Human-Robot Interaction architecture for interactive and lively social robots

por

Enrique Fernández Rodicio

Tesis depositada en cumplimiento parcial de los requisitos para el grado de Doctor en

Ingeniería Eléctrica, Electrónica y Automática

Universidad Carlos III de Madrid

Directores:

Miguel Ángel Salichs Sánchez-Caballero
Álvaro Castro González

Tutor:

Miguel Ángel Salichs Sánchez-Caballero

Julio 2021

A mi familia.

AGRADECIMIENTOS

Esta tesis supone la culminación de cinco años de mi vida académica que me han proporcionado un sinfín de buenos momentos (y por que negarlo, también una cantidad considerable de trabajo de vez en cuando). Aunque tengo la esperanza de que cerrar esta etapa de mi formación no suponga el final de mi vinculación con el mundo de la universidad y la investigación, es innegable que este es uno de los mayores hitos que he conseguido en mi vida hasta este momento. Es algo que de ningúna manera podría haber conseguido sin la ayuda y el apoyo de una serie de personas a las que tengo mucho que agradecer.

Me gustaría empezar por agradecer a mis directores de tesis, Miguel Ángel y Álvaro. Fue Miguel Ángel el que me abrió las puertas del grupo de investigación que ha pasado a convertirse casi en parte de mi familia durante los últimos cinco años. Cuando yo era alumno del máster en robótica, fueron las clases de Miguel Ángel las que me hicieron decidirme a orientar mi formación hacia la robótica social, y es una decisión que todavía me alegro de haber tomado. Además, fue él quién me proporcionó la oportunidad de enfrentarme al desafío que ha sido completar esta tesis doctoral. Álvaro por otra parte ha sido uno de mis mayores apoyos durante mi estancia en el Grupo de Róbotica Social, ya desde el primer día que coincidimos en las Shangai Lectures. Una buena parte del mérito del trabajo que se presenta en este documento es suya, por la dedicación que ha puesto a ayudarme en cada paso del camino. Es ingente la cantidad de cosas que he tenido la fortuna de aprender de él, tanto a nivel profesional como personal. Siempre es capaz de sacar un rato para ayudarme con las dudas y problemas que me he ido encontrando durante estos cinco años (y eso que según se acercaba el final de esta tesis, he ido siendo progresivamente más y más pesado). Gracias a los dos.

Y que decir de mi familia. No hay espacio suficiente en esta sección para mencionar todas las cosas por las que les tengo que estar agradecido a mis padres y mis hermanas, Sabela y Marta. Creo que es una obviedad, pero no sería ni la mitad de persona que soy hoy en día si no fuese por ellos, y por la educación que me han dado. Siempre han estado ahí para apoyarme en los buenos y malos momentos, han sabído perdonar los errores que he ido cometiendo a lo largo de los años, y me han apoyado en las diferentes decisiones que me han llevado desde Galicia hasta Madrid. Su ejemplo y esfuerzo han sido algo que siempre me ha servido de inspiración cuando me he tenido que enfrentar a los diferentes desafíos que la vida me ha ido poniendo delante. Creo que no lo digo todo lo que debería, pero gracias por todo. También me gustaría dar las gracias a mis abuelos, tíos y tías (Geni, Iago, Liza, Eloy, Tomás, Sofía, Isa, Rafa, Susana...), y a mis primos y primas (Xabi, Álvaro, Lara, Lucía, Javi, Alberto, Natalia...). Gracias familia.

No me puedo olvidar de toda la gente que forma parte del Grupo de Robótica Social de la Universidad Carlos III. Gente como Jose Carlos, Juanjo, Sara, Marcos, Esther, María, Fer, Carlos, Javi, y tantas otras personas que han pasado por el grupo durante mi tiempo aquí. Son incontables las veces que sus buenos consejos me han ayudado a avanzar en mi trabajo, y creo que he conseguido aprender algo de todos ellos. También se merecen que les de las gracias Sonia y Edu, quienes han tenido que sufrir mis limitadas aptitudes para la organización durante más de un trámite, y aún así siempre consiguieron que todo saliese adelante. También me gustaría agradecer a mis compañeros

del máster y de Asrob: David, Carlos, Rober, Raúl, Javi, Clara, Manu... Son la gente que ayudó a que creciera en mí la pasión por la robótica, a través de incontables conversaciones y noches de cervezas. No puedo olvidarme tampoco de la gente del departamento y de los doctorandos que han sido mis compañeros de alegrías y penurias durante la realización de mi tesis. Mención especial merecen mis compañeros del despacho 1.3.B15, Jonathan y Edwin, que han tenido que soportar una buena cantidad de discursiones, experimentos, revisiones... Por último, pero no por ello menos importante, tengo que darle las gracias a los que han sido mis compañeros y amigos durante mi formación universitaria: Adri, Gaudul, Elia, Adela, Sergio, Guille, Iván, Alber, Santi, Cris, Gonzalo, y tantos otros. Todos vosotros ayudasteis a que un recién llegado a la universidad y a Madrid se sintiese como en casa. Muchas gracias a todos.

Venirme a Madrid dejando todo atrás fue un gran cambio en mi vida, pero la gente que tuve la fortuna de conocer en el colegio mayor hicieron este cambio mucho más sencillo. Desde los veteranos que me sirvieron de ejemplo, como Chechu, Pablo, Reda, Esther, o Javi, pasando por la gran cantidad de amigos y amigas con las que tanto he compartido, como Óscar, Loli, Nacho, María, Laura, Eugenia, Omar, Garibay, Máximo, Elvira, y tantas otras personas a las que les estoy muy agradecido. Y por supuesto, a Amadeo, Javi, y Jaime, con quienes he tenido la fortuna de convivir a diario durante los últimos diez años, los mejores compañeros de piso que uno podría pedir. Son tantas historias por las que hemos pasado juntos, y espero que aún nos queden las mejores por delante.

Morriña es esa preciosa palabra que hace referencia a la tristeza y la nostalgia cuando uno se encuentra lejos de su hogar. Pero desde hace ya unos años, ha tomado un nuevo significado para mí: amistad. Fue en 2015 cuando, como celtista en el exilio, decidí acercarme hasta la peña Morriña Celeste a compartir las alegrías (pocas, para que engañarnos) y penurias (demasiadas) que este equipo me hace pasar. Y allí me encontré con un maravilloso grupo de gente que me acogió desde el primer día como si fuese uno más. Han paasado ya casi seis años en los que he coleccionado una gran cantidad de fantasticos recuerdos con gente que se ha convertido en parte muy importante de mi vida: Dioni, Noa, Charlie, Juanri, Alfonso, Tere, Fer, Mario, Iago, Tereixa, Clara, Bastida, Besada, Laura, Alex, Pili, Olalla, Xoan... Moitas grazas a todos e todas, e ¡ala Celta!

No puedo olvidarme de los investigadores con los que trabajé durante mi estancia en Edinburgo: Christian, Andrea, Iannis, Oliver, y tantos otros que me acogieron como si llevase allí toda la vida, y ayudaron a que esta experiencia haya sido un auténtico éxito. Thank you all.

Por último, me gustaría agradecer a la Universidad Carlos III y todas las personas que trabajan en ella por la formación que he recibido, y a las distintas organizaciónes y ministerios que han ayudado a financiar esta investigación y que han permitido que haya llegado hasta aquí.

A todos y todas, muchisimas gracias.

CONTENIDOS PUBLICADOS Y PRESENTADOS

Publicaciones en revistas

1. **E. Fernández-Rodicio**, A. Castro-González, F. Alonso, M. Maroto, M.A. Salichs. Modelling Multimodal Dialogues for Social Robots Using Communicative Acts. *Journal of Sensors*, 20 (12), 3440. 2020. DOI: [10.3390/s20123440]. **Q2** en [Electrical and Electronic Engineering].

   ***El material de esta fuente ha sido parcialmente incluido en esta tesis en el Capítulo 3. El material de esta fuente incluido en la tesis no está señalado por medios tipográficos ni referencias.*

2. M.A. Salichs, A. Castro-González, E. Salichs, **E. Fernández-Rodicio**, M. Maroto, J.J. Gamboa, S. Marques-Villarroya, Jose C. Castillo, M. Malfaz, F. Alonso. Mini: A New Social Robot for the Elderly. *International Journal of Social Robotics*, 12, 1231-1249. 2020. DOI: [https://doi.org/10.1007/s12369-020-00687-0]. **Q1** en [General Computer Science].

   ***El material de esta fuente ha sido parcialmente incluido en esta tesis en los Capítulos 2, 3, 4, and 5. El material de esta fuente incluido en la tesis no está señalado por medios tipográficos y una referencia explícita.*

Publicaciones en congresos

1. **E. Fernández-Rodicio**, A. Castro-González, J.J. Gamboa, M.A. Salichs. Perception of a Social Robot's Mood Based on Different Types of Motions and Coloured Heart. International Conference on Social Robotics, Golden, Colorado (USA), November 2020.

   ***El material de esta fuente ha sido parcialmente incluido en esta tesis en el Capítulo 4. El material de esta fuente incluido en la tesis no está señalado por medios tipográficos ni referencias.*

2. **E. Fernández-Rodicio**, A. Castro-González, S. Marques-Villarroya, M.A. Salichs. Gesture Management in the Robot Mini. Workshop on Expressivity for sustained Human-Robot Interaction. International Conference on Human-Robot Interaction, Daegu (South Korea), March 2019.

   ***El material de esta fuente ha sido parcialmente incluido en esta tesis en el Capítulo 4. El material de esta fuente incluido en la tesis no está señalado por medios tipográficos ni referencias.*

3. **E. Fernández-Rodicio**, A. Castro-González, Jose C. Castillo, F. Alonso, M.A. Salichs. Composable Multimodal Dialogues based on Communicative Acts. International Conference on Social Robotics, Qingdao (China), November 2018.

   ***El material de esta fuente ha sido parcialmente incluido en esta tesis en el Capítulo 3. El material de esta fuente incluido en la tesis no está señalado por medios tipográficos ni referencias.*

4. E. Salichs, **E. Fernández-Rodicio**, Jose C. Castillo, A. Castro-González, M. Malfaz, M.A. Salichs. A social Robot Assisting in Cognitive Stimulation Therapy. Advances in Practical Applications of Agents, Multi-Agent Systems and Complexity: The PAAMS Collection, June 2018. Lecture Notes in Computer Science, vol 10978. Springer, Cham. https://doi.org/10.1007/978-3-319-94580-4_35.

**El material de esta fuente ha sido parcialmente incluido en esta tesis en los Capítulos 2 and 3. El material de esta fuente incluido en la tesis no está señalado por medios tipográficos ni referencias.*

Resumen

La sociedad está experimentando un proceso de envejecimiento que puede provocar un desequilibrio entre la población en edad de trabajar y aquella fuera del mercado de trabajo. Una de las soluciones a este problema que se están considerando hoy en día es la introducción de robots en multiples sectores, incluyendo el de servicios. Sin embargo, para que esto sea una solución viable, estos robots necesitan ser capaces de interactuar con personas de manera satisfactoria, entre otras habilidades. En el contexto de la aplicación de robots sociales al cuidado de mayores, esta tesis busca proporcionar a un robot social las habilidades necesarias para crear interacciones entre humanos y robots que sean naturales. En concreto, esta tesis se centra en tres problemas que deben ser solucionados: (i) el modelado de interacciones entre humanos y robots; (ii) equipar a un robot social con las capacidades expresivas necesarias para una comunicación satisfactoria; y (iii) darle al robot una apariencia vivaz.

La solución al problema de modelado de diálogos presentada en esta tesis propone diseñar estos diálogos como una secuencia de elementos atómicos llamados Actos Comunicativos (CAs, por sus siglas en inglés). Se pueden parametrizar en tiempo de ejecución para completar diferentes objetivos comunicativos, y están equipados con mecanismos para manejar algunas de las imprecisiones que pueden aparecer durante interacciones. Estos CAs han sido identificados a partir de la combinación de dos dimensiones: iniciativa (si la tiene el robot o el usuario) e intención (si se pretende obtener o proporcionar información). Estos CAs pueden ser combinados siguiendo una estructura jerárquica para crear estructuras mas complejas que sean reutilizables. Esto simplifica el proceso para crear nuevas interacciones, permitiendo a los desarrolladores centrarse exclusivamente en diseñar el flujo del diálogo, sin tener que preocuparse de reimplementar otras funcionalidades que tienen que estar presentes en todas las interacciones (como el manejo de errores, por ejemplo).

La expresividad del robot está basada en el uso de una librería de gestos, o expresiones, multimodales predefinidos, modelados como estructuras similares a máquinas de estados. El módulo que controla la expresividad recibe peticiones para realizar dichas expresiones, planifica su ejecución para evitar cualquier conflicto que pueda aparecer, las carga, y comprueba que su ejecución se complete sin problemas. El sistema es capaz también de generar estas expresiones en tiempo de ejecución a partir de una lista de acciones unimodales (como decir una frase, o mover una articulación). Una de las características más importantes de la arquitectura de expresividad propuesta es la integración de una serie de métodos de modulación que pueden ser usados para modificar los gestos del robot en tiempo de ejecución. Esto permite al robot adaptar estas expresiones en base a circumstancias particulares (aumentando al mismo tiempo la variabilidad de la expresividad del

robot), y usar un número limitado de gestos para mostrar diferentes estados internos (como el estado emocional).

Teniendo en cuenta que ser reconocido como un ser vivo es un requisito para poder participar en interacciones sociales, que un robot social muestre una apariencia de vivacidad es un factor clave en interacciones entre humanos y robots. Para ello, esta tesis propone dos soluciones. El primer método genera acciones a través de las diferentes interfaces del robot a intervalos. La frecuencia e intensidad de estas acciones están definidas en base a una señal que representa el pulso del robot. Dicha señal puede adaptarse al contexto de la interacción o al estado interno del robot. El segundo método enriquece las interacciones verbales entre el robot y el usuario prediciendo los gestos no verbales más apropiados en base al contenido del diálogo y a la intención comunicativa del robot. Un modelo basado en aprendizaje automático recibe la transcripción del mensaje verbal del robot, predice los gestos que deberían acompañarlo, y los sincroniza para que cada gesto empiece en el momento preciso. Este modelo se ha desarrollado usando una combinación de un encoder diseñado con una red neuronal *Long-Short Term Memory*, y un *Conditional Random Field* para predecir la secuencia de gestos que deben acompañar a la frase del robot.

Todos los elementos presentados conforman el núcleo de una arquitectura de interacción humano-robot modular que ha sido integrada en múltiples plataformas, y probada bajo diferentes condiciones. El objetivo central de esta tesis es contribuir al área de interacción humano-robot con una nueva solución que es modular e independiente de la plataforma robótica, y que se centra en proporcionar a los desarrolladores las herramientas necesarias para desarrollar aplicaciones que requieran interacciones con personas.

Abstract

Society is experiencing a series of demographic changes that can result in an unbalance between the active working and non-working age populations. One of the solutions considered to mitigate this problem is the inclusion of robots in multiple sectors, including the service sector. But for this to be a viable solution, among other features, robots need to be able to interact with humans successfully. This thesis seeks to endow a social robot with the abilities required for a natural human-robot interactions. The main objective is to contribute to the body of knowledge on the area of Human-Robot Interaction with a new, platform-independent, modular approach that focuses on giving roboticists the tools required to develop applications that involve interactions with humans. In particular, this thesis focuses on three problems that need to be addressed: (i) modelling interactions between a robot and an user; (ii) endow the robot with the expressive capabilities required for a successful communication; and (iii) endow the robot with a lively appearance.

The approach to dialogue modelling presented in this thesis proposes to model dialogues as a sequence of atomic interaction units, called Communicative Acts, or CAs. They can be parametrized in runtime to achieve different communicative goals, and are endowed with mechanisms oriented to solve some of the uncertainties related to interaction. Two dimensions have been used to identify the required CAs: initiative (the robot or the user), and intention (either retrieve information or to convey it). These basic CAs can be combined in a hierarchical manner to create more re-usable complex structures. This approach simplifies the creation of new interactions, by allowing developers to focus exclusively on designing the flow of the dialogue, without having to re-implement functionalities that are common to all dialogues (like error handling, for example).

The expressiveness of the robot is based on the use of a library of predefined multimodal gestures, or expressions, modelled as state machines. The module managing the expressiveness receives requests for performing gestures, schedules their execution in order to avoid any possible conflict that might arise, loads them, and ensures that their execution goes without problems. The proposed approach is also able to generate expressions in runtime based on a list of unimodal actions (an utterance, the motion of a limb, etc...). One of the key features of the proposed expressiveness management approach is the integration of a series of modulation techniques that can be used to modify the robot's expressions in runtime. This would allow the robot to adapt them to the particularities of a given situation (which would also increase the variability of the robot expressiveness), and to display different internal states with the same expressions.

Considering that being recognized as a living being is a requirement for engaging in social encounters, the perception of a social robot as a living entity is a key requirement to foster human-robot interactions. In this dissertation, two approaches have been proposed. The first method generates actions for the different interfaces of the robot at certain intervals. The frequency and intensity of these actions are defined by a signal that represents the pulse of the robot, which can be adapted to the context of the interaction or the internal state of the robot. The second method enhances the robot's utterance by predicting the appropriate non-verbal expressions that should accompany them, according to the content of the robot's message, as well as its communicative intention. A deep learning model receives the transcription of the robot's utterances, predicts which expressions should accompany it, and synchronizes them, so each gesture selected starts at the appropriate time. The model has been developed using a combination of a Long-Short Term Memory network-based encoder and a Conditional Random Field for generating a sequence of gestures that are combined with the robot's utterance.

All the elements presented above conform the core of a modular Human-Robot Interaction architecture that has been integrated in multiple platforms, and tested under different conditions.

Contents

List of Tables

List of Figures

Acronym list

AI	Artificial Intelligence
ANCOVA	Analysis of covariance
API	Application Programming Interface
ASR	Automatic Speech Recognition
BPTT	Backpropagation Through Time
BERT	Bi-directional Encoder Representations from Transformers
BML	Behaviour Markup Language
CA	Communicative Act
CCA	Complex Communicative Act
CHMM	Coupled Hidden Markov Models
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRF	Conditional Random Field
CEC	Constant Error Carrousel
DCNF	Deep Conditional Neural Field
DMS	Decision Making System
DQN	Deep Q-Networks
ELMo	Embeddings from Language Models
FiLM	Feature-wise Linear Modulation
FML	Function Markup Language
GAN	Generative Adversarial Network
GIF	Graphics Interchange Format
GPLVM	Gaussian Process Latent Variable Model
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HRI	Human-Robot Interaction

LED	Light-Emitting Diode
LSTM	Long Short Term Memory
MFCC	Mel Frequency Cepstrum Coefficients
MLP	Multilayer Perceptron
NLG	Natural Language Generation
NLP	Natural Language Processing
POMDP	Partially-Observable Markov Decision Process
PoS	Part-of-Speech
RAM	Random Access Memory
RDF	Resource Description Framework
RGB-D	Red-Green-Blue-Depth
ROS	Robot Operating System
RoSAS	Robotic Social Attributes Scale (ROSAS)
TCN	Temporal Convolution Network
TFT	Thin-film Transistor
TTS	Text-To-Speech
URL	Uniform Resource Locator
XML	Extensible Markup Language

CHAPTER 1

Introduction

1.1 Motivation

The field of robotics has experienced a remarkable growth in recent years. According to an study conducted by Oxford Economics in 2019 [1], the number of robots has multiplied three-fold in the last 20 years, reaching 2.25 million, and observed trends point towards an even faster growth in the near future. By the year 2030, the number of active robots in the world is expected to be around 20 million, 14 of those on China alone. Figure 1.1 shows the increase of new installations of robotic applications between 2000 and 2016. Among the different taxonomies that can be used to classify robots [4], focusing on the field in which those robots are going to be applied, we find a distinction between two types of robots: industrial and non-industrial robots. The former can be classified depending on the task they are built for (welding, palletizing, assembly, etc...) while, for the latter, two subgroups can be identified: (i) robots for production of goods, and (ii) service provider robots. While industrial robots are already common in factories all over the world, the usage of robots in the service sector is starting to gain traction. Data provided by the International Federation of Robotics shows that the market for professional service robots grew a 61% between 2017 and 2018, primarily due to the expansion of logistics robotic systems. In the same period of time, personal robots sales saw a 59% increase. In this last category, cleaning robots accounted for 67% of total sales, with toy robots in second place. While robots designed for helping the elderly and people needed of assistance currently represent a small percentage of total personal robots sales (1.3%), this percentage is expected to grow on average a 29% per year between 2019 and 2022, which shows their promise.

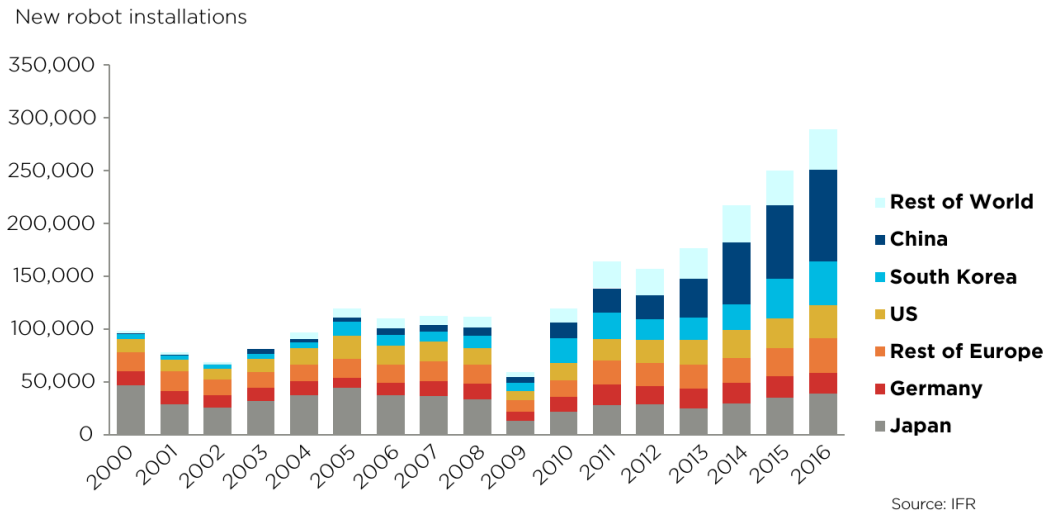


Figure 1.1: New robot installations by country in the 21^{st} century. Figure extracted from the Oxford Economics report[1].

There are several factors that can explain this growing interest in robotics. Besides the economic factors (for example, robots are starting to become more cost effective than humans[1]), there is also a demographic factor that plays a part in this expansion. According to information provided by the World Health Organization, in 2015, 12% of the population was over 60 years old. This percentage is expected to raise to a 20% by 2050. The pace of population ageing is also increasing drastically. While it took nearly 150 years for an 10% increase in population over 60 years old in France (from 10% of the population to 20%), is going to take slightly more than 20 years for countries like India, China, or Brazil to see the same variation. This trend will cause important economic [5] and social [6] effects on our society. On one hand, this ageing process will bring a change in the balance between the sizes of the population of working age and the population of not-working age. Due to the current economic structure of developed countries, where the benefits of retired persons are covered by people in working age, a decreasing workforce will have to support an increasing elder population. This shortage of workers could be palliated with an increase on retirement age, and the inclusion of the older sectors of the population into the labour market, which in turn will require businesses to attend the needs and capacities of older employees. On top of this, an ageing population has an associated increase on the need for healthcare services and a change on the distribution of public spending, with an increase on healthcare financing. In part, this is due to an expected increase on the size of the population suffering from diseases commonly associated with old age (for example, Parkinson's disease or Alzheimer's disease) [7], which will bring an increase of the demand for adult primary care.

The combination of the growing interest on robotics and the problems that society will experience in the near future suggests that the development of robotic platforms that can perform tasks in the service sector, both in professional environments (public relations, commerce, etc...) and as providers of personal care (companion robots, healthcare assistance, etc...), can help to mitigate the challenges presented above. But for the integration of robots in the service sector to be a viable solution, robots will require to be endowed with a series of capabilities that give them the ability to interact with humans in a natural way. In order to be considered as natural, an interaction needs to abide by the social rules enforced in the domains in which the robot will be inserted, and meet the expectations that the humans in said domains have for human-human interactions. The subfield of robotics that focus in providing a solution to this problem is known as social robotics.

Cynthia Breazeal [8] defined a social robot as "an autonomous robot that can communicate with humans in accordance to a social model that is applied by the human observers". A similar definition was proposed by Bartneck et al. [9]: "Autonomous or semi-autonomous robot able to interact and communicate with humans according to the behavioural norms expected by these humans". Based on the degree to which the robot adheres to the social model attributed by the humans interacting with the robot, Breazeal categorizes social robots into the following classes, from lower to higher social intelligence:

1. **Socially evocative:** robots that encourage users to attribute them social responsiveness, without actually possessing it.
2. **Social interfaces:** robots that can use human-like social cues and interaction modalities, but in a predefined and often reflexive social behaviour.
3. **Socially receptive:** robots that can learn from human social cues and use them to modify their behaviour, without being proactive and have its own communicative goals.
4. **Sociable:** robots that are socially participative, posses their own interaction goals, and can proactively engage people in communication.

Based on the definitions given above, what characterizes a social robot is how a human observer perceives it in the context of a given social model. But the concept of social model is completely subjective, as it is created by said human observers. Although individuals belonging to the same demographic group might have a similar expectation of how a robot should behave, this might

not be necessarily true for individuals from different groups. Also, the social rules defined for human-human interactions are not universal, but instead depend on the situation and the context. For example, how persons interact in a professional environment can be completely different from how two persons with a personal relationship interact. Because of this, the design of a social robot is usually dependant on the field in which the robot is going to be used [10, 11, 12]. Still, there are a series of common features that have been deemed necessary in the studies presented above. For example, multimodality (the ability to use multiple communication channels, either separately or in combination) is usually desired, both for communicating with the robot and for the robot to convey messages. Also, social robots need to adapt to the user and their surroundings, and show a large range of conversational responses to multiple events (something that could be achieved through this adaptation). However, not all the participants in the studies cited above agreed on if these responses have to be human-like, or can be closer to the response expected from animals. Another feature that people find important in a social robot is the capability to express affect states (either mood or emotions) and to recognize those same states on the humans. Part of these features can be associated to the cognitive aspects of the interaction (making decisions about how the robot should react given the internal state of the robot, the previous action of the human with whom it is interacting, and the knowledge that can be extracted from the environment), while others are tied to the expressiveness capabilities of the robot. The latter will be the main focus of this dissertation.

1.2 The problem

As seen in this chapter, for robots to be able to serve as companions, assistants, or providers of services, they need to possess a series of interaction capabilities that allow them to comply with the set of rules and behaviours that society has accorded for each of these positions. Among other features, robots must possess a high level of adaptability, social and emotional intelligence, and expressiveness, in order to provide an experience as satisfactory as possible. But achieving a satisfactory interaction presents a set of problems that need to be solved. The work developed in this thesis will focus on three of these problems: 1) how to control the flow of the dialogue and decide what the response of the robot should be (dialogue management), 2) how the robot should convey information to the user (expressivity management), and 3) how the robot can transmit an appearance of being a living being (animacy expression).

1.2.1 Dialogue Management

In robotic architectures, the module that endows a social robot with the ability to interact with humans is usually named Dialogue System. A Dialogue System is a computational agent designed to converse with a human using everyday language. While starting as text-based applications for Human-Computer interaction, soon they evolved to include other means of interaction. In particular, Spoken Dialogue systems have beenthe ones attracting more attention, as spoken language is a cornerstone of a person's daily interactions with the world, due to the fact that speech can convey complex ideas in a highly efficient manner [13]. Nowadays, dialogue systems have become a part of our society with the proliferation of portable computer systems and smartphones, and the emergence of voice assistants [14] like Alexa, or Cortana. These systems manage a wide range of tasks, from speech recognition or perceiving stimuli coming from the environment, to modelling how a response should be given so it conveys the desired information. Inside a Dialogue System, the element that manages the flow of the interaction and decides on the next action of the system based on previous steps of the dialogue and the knowledge base of the system is the Dialogue Manager.

The Dialogue Manager is the core element in a Dialogue System [15, 16, 17]. It is in charge of maintaining the representation for the robot's dialogues. The Dialogue Manager receives information coming from the input processing modules of the system, and advances the dialogue accordingly, selecting the most appropriate system actions at every moment. In a sense, the Dialogue Manager acts as the brain of a dialogue System. Thus, all the features of a social robot that give it the ability of react appropriately to the user's communicative actions and assure that the conversational goals are fulfilled are usually integrated in this module. In this area, one of the questions that this dissertation will aim at answering is how to model dialogues for a social robot in a way that makes them easy to design, and how to endow the robot with the ability to manage and complete said dialogues.

1.2.2 Expressiveness in Robotics

According to the Cambridge dictionary, expressiveness is defined as *"the state of showing what someone thinks or feels"*. This implies that there is a communicative component to expressiveness, as its final objective is to convey a particular message. But expressions are not only a tool to communicate information to other humans, they also have an effect on how those persons perceive us. For example, studies [18] suggest that individuals who are non-verbally skilled and extroverted, and thus display outwardly focused and fluid expressive behaviours, tend to make a better impression on people.

This means that the development of expressiveness features for a social robot has to be addressed from two different perspectives. On one hand, the robot has to be able to express itself in a way that is understandable and feels appropriate to the user. In order to achieve this, roboticists have to design models that can be used to represent multimodal expressions for social robots, and also to design strategies for conveying said expressions. These strategies should be able to adapt the robot's expressiveness to the particularities of the human interacting with it and the characteristics of the environment, while maintaining the integrity of the message that has to be conveyed. On the other hand, it is imperative to consider the effect that the robot's expressions have over the persons that will interact with it. The goal is to design this expressiveness in a way that helps the robot to be perceived as a social agent. This can be achieved through the development of methods for modulating the expressions, so they can reflect the internal state of the robot at every moment, and also be adapted to the particular circumstances of an interaction.

Endowing a social robot with the proper expressiveness that combine the two perspectives presented above will help the robot to provide more engaging interactions, help to create emotional bonds between the robot and the human user, and overall improve the ability of the robot to fulfil the role for which it has been designed.

1.2.3 Animacy and social robots

The *biophilia hypothesis* [19, 20] proposes that *"humans have an innate tendency to focus on and affiliate with life forms and life-like processes"*. Based on this theory, it could be possible to argue that robots that display life-like characteristics will be easier to accept for humans than those that clearly convey a machine-like appearance.

This premise has been supported by research in the field of psychology. In 2009, Wheatly et al. [21] proposed that animacy (the quality that an entity has of being recognized as a living being) is a keystone for social interactions, and the fact that a human recognizes his/her interaction partner as a living being is a prerequisite for other high level social functions, like the establishment of communication. The degree to which this animacy has to be achieved in social robotics is still an open question, though. For example, Melson et al. [22], when studying the use of robotic pets as companions and therapeutical tools, observed that persons would recognize the robotic dog as a machine, but still assign it some attributes associated with living beings. Thus, they defended the use of a separate category when defining robots: "sort of alive".

A significant amount of work has been dedicated in the field of social robotics to how this animacy can be given to a robot, and what features result relevant. Authors like Bartneck et al. [23] proposed a combination of vibrant facial expressions and human-like physical features as a method for achieving a high animacy perception. Nakayama et al. [24] found that robots with abstract shapes and simple mechanisms can convey the impression thanks to its movement. Castro et al. [25] also evaluated the use of motion, but also the effect of bodily appearance.

The work developed in this thesis seeks to contribute to the body of knowledge on animacy in social robotics with the development of two methods for endowing a robot with a lively appearance.

1.3 Objectives

This thesis focuses on how to provide a social robot with the capabilities that are required for achieving a natural interaction with users. In this context, a *natural* interaction is defined as an interaction that abides by the social rules that an agent in that environment is expected to follow, and that meets the expectations that the human interacting with the robot might have. Thus, the main objective of this dissertation is **to endow a social robot with the necessary communicative abilities in order to provide a interaction that feels satisfactory to a human user**. The proposed solution will be integrated in several robotic platforms, and evaluated through user studies that will validate its usefulness. Although the work presented in this dissertation has been integrated in robotic platforms developed with a specific field of application in mind (companion robots for older adults that suffer from mild cases cognitive impairment), the desired solution should not be constrained to a specific task, and instead be used in multiple domains.

To achieve this main objective, a series of subgoals have been defined:

1. The proposed system has to be easy for developers to use. The objective is to achieve an interaction architecture that can manage the interaction-related aspects of communication in a way that feels natural to a human observer, while simplifying the design of new dialogues in different domains.
2. The proposed system has to be highly modular, in order to simplify the process of upgrading or replacing the different components of the architecture without affecting the rest of the system.
3. The proposed system has to be application independent, and it should be possible to integrate it in robotic architectures for multiple domains without the need for major adjustments.

4. The overall performance of the proposed system has to allow an interaction that meets the temporal constraints that apply to human-human interactions. Thus, the reaction time of the system will be used as a key benchmark feature.

5. The proposed system has to be flexible enough to adapt to the particularities of different users, in order to introduce as little friction as possible in the interactions between the human and the robot.

6. The final architecture has to be validated through an extensive subjective evaluation. The participation in this evaluation will not be restricted to users belonging to the field of application of the robots used in the study (older adults).

7. The proposed architecture will have to be installed and run in a robotic platform. Due to the limited hardware resources (mainly, the RAM memory and the processing power) that robots have, and the fact that other modules have to be running in the robot as well, it is necessary to evaluate the proposed architecture's performance.

Besides these general objectives, specific subgoals for each of the three aspects of human-robot interaction considered in this thesis will be presented in their respective chapters.

1.4 Overview of the document

The work in this thesis has been divided in three main parts that correspond to three areas of human-robot interactions: i) dialogue management, ii) expressivity management, and iii) animacy expression. Next, the contents of the different chapters in this document are presented in depth:

- **Chapter 2:** This chapter presents the robotic platforms in which the tools developed in this thesis have been integrated. It starts with a brief description of the hardware of each robot, and then presents the software architecture, which is common to all platforms.

- **Chapter 3:** This chapter focuses on dialogue management in social robots. First, a brief introduction to the challenges that have to be solved in this field is presented, as well as the research objectives that we have defined. Then, a review of the state of the art in dialogue management is conducted, divided in the different approaches that can be followed for solving this problem. The final part of this section shows a comparative analysis of the

works reviewed, focusing on highlighting the voids in the state of the art that the proposed approach is trying to fill. Next, the chapter presents the concepts of dialogue management in human-human interactions that serve as the theoretical foundation for the proposed dialogue management system. Then, the technical aspects of the implementation of the system developed are presented, along with its integration into the robot's software architecture. After, an evaluation of the proposed approach is conducted in order to assess its performance. Finally, I review the conclusions extracted from the work conducted in dialogue management, and present the main contributions of this research. The main limitations of the proposed dialogue management approach are presented, along with a series of possible future lines of work.

- **Chapter 4:** In this chapter, the expressiveness aspect of human-robot interactions is presented. First, an introduction to expressiveness in social robotics and the definition of the expressiveness-related objectives of this thesis are presented. Then, I conduct a review of the state of the art of expressiveness architectures for social robots. The next section introduces the proposed expressiveness approach, from a theoretical point of view. Then, the technical aspects of the proposed solution are presented, as well as the integration of this solution in the general architecture of the robot. After, I present a series of studies aimed at evaluating how users perceive the expressiveness of our robots when using the proposed system. This chapter is closed with a compilation of the conclusions extracted regarding the expressiveness of social robots, an evaluation of the goals defined beforehand, the main contributions of the work developed in this particular area, and also the limitations of the proposed system and future lines of work.

- **Chapter 5:** The last area of human-robot interaction studied in this thesis is presented in this chapter: how can robots convey a liveliness appearance. Following the structure of chapters 3 and 4, I begin with an introduction to the problem in need of solving, the goals that have to be achieved, and a review of the current state of the art in this field of robotics. The core section of this chapter introduces the proposed liveliness generation methods, as well as its integration in the robotic platforms.Next, I present the evaluations performed to test the performance of the proposed liveliness generation approaches. Finally, the conclusions for my research on liveliness expression are presented, highlighting the main contributions of this work. This last section also presents the limitations of the work developed, as well as possible future works.

- **Chapter 6:** In this last chapter, I summarize the general conclusions extracted from my research, and provide an overall evaluation of the work presented in this dissertation.

CHAPTER 2

Framework of the thesis

2.1 Introduction

The embodiment of a communicative agent plays an important role on how its interaction capabilities are designed. The strategy followed for managing the dialogues between the robot and the agent has to take into account the type of messages the agent can convey, and also the different sources of information that can be received. The solution proposed for dealing with these platform-related particularities was the development of a modular architecture that tries to separate those aspects of a robot's software architecture that are platform-specific from those that are more generic. Each part of the architecture is compartmentalized, and the communication is done through standardized interfaces. This simplifies the process of upgrading or replacing individual components, or adding new skills. In order to integrate the architecture in a new robot, only the modules in contact with the robot's hardware need to be modified, while the rest of the modules can remain unchanged.

This section introduces the robotic platforms in which the proposed interaction architecture has been implemented and evaluated, along with a review of their hardware features. This includes a brief presentation of the research group and some of their research projects. Next, the chapter presents the software architecture that has been integrated in these robotic platforms. Finally, this chapter will be closed with a summary of the key concepts discussed.

2.2 Robotic platforms

All the work developed for this thesis has been conducted in the Robotics Lab of the Carlos III University of Madrid, inside the Social Robotics Group. This research group focuses on developing social robots and their applications. The main research lines include robotic perception, decision making, dialogue management, expressiveness management, Human-Robot Interaction, and cognitive stimulation. The research group has taken part in research projects with Spanish and European companies and institutions. Among the different projects, the following can be highlighted:

- **Development of social robots for assisting older adults with cognitive impairment (ROBSEN):** This project had the goal of developing a social robot for assisting older adults that suffer from mild cases of cognitive impairment. In this case, the robot helps the caregiver (without replacing him/her), and provides assistance to the patient in four scenarios: entertainment, stimulation, personal assistance, and security and safety.
- **Social robots for physical, cognitive, and affective stimulation for older adults (ROSAS):** This project focuses on the use of robotic platforms to perform cognitive, physical, and affective stimulation therapies with older adults.
- **MOnarCH (Multi-Robot Cognitive Systems Operating in Hospitals):** This research project took place between 2013 and 2016, and focused on using social robots to interact with children, staff and visitors in the pediatric unit at the Portuguese Oncology Institute of Lisbon. A new social robot was specifically designed and built for this project. The work conducted in the Social Robotics Lab focused on developing the HRI capabilities of the robot.

Since 2005, multiple robotic platforms have been developed, and/or used for research purposes in the Social Robotics Lab. The first robot that was completely designed and built inside the group was Maggie [26], a personal social robot designed as a research platform for studying Human-Robot Interaction (HRI), robot cognition, and robot autonomy. Another platform used in the research group is Mbot [27], a child-sized mobile robot for interacting with paediatric patients in an oncological hospital. Both robots are shown in Figure 2.1

In the frame of the more recent research projects, two new platforms designed to assist older adults that suffer from mild cognitive impairment were developed: Mini and Gero. These are the robotic platforms in which the interaction architecture designed for this thesis has been integrated and evaluated. They will be presented in depth in the following subsections.

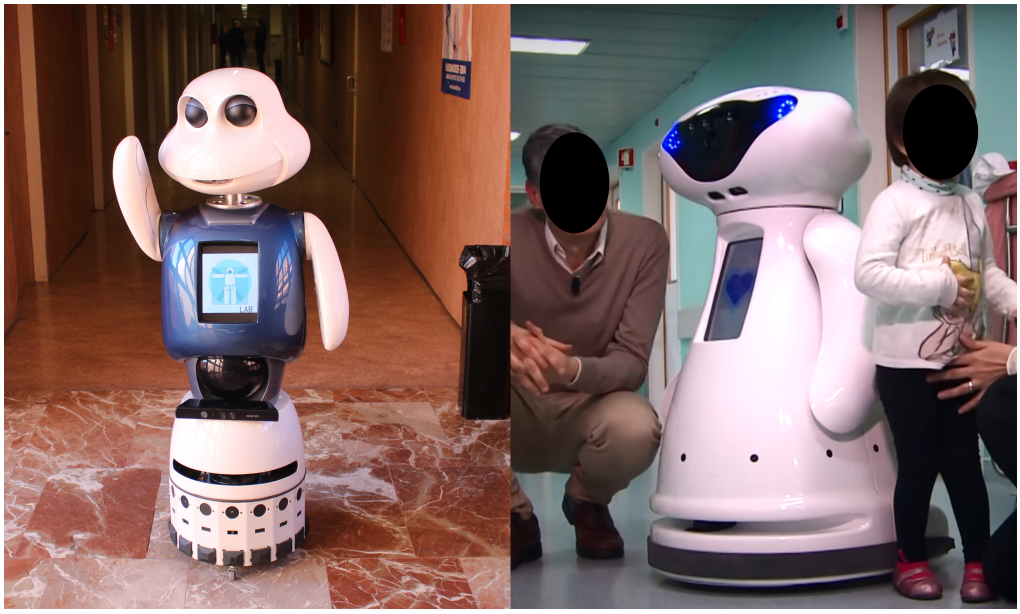


Figure 2.1: Maggie, a social robot designed as a research platform (left), and Mbot (right), a robot for entertaining children in a hospital (source: [2]).

2.2.1 Mini

Mini [28], shown in Figure 2.2, is a social robot designed to help older adults and their caregivers in their daily life activities, either in nursing facilities or in the users' houses. It was conceived as a tool that can be used by physicians, instead of replacing them. Users can play games with Mini, request different multimedia content (movies, photos, music...), ask for a weather report or the news, and also complete cognitive stimulation exercises and therapies.

Modelled after the robot Maggie, Mini has an anthropomorphic shape, although its appearance is closer to that of a cartoon, being soft and squashy. Unlike Maggie, Mini was designed as a tabletop robot, based on the feedback provided by experts in the assistance care field. The internal skeleton of the robot is 3D printed, and houses the microcontroller that manages the sensors and actuators. The box at the bottom of the robot contains the computer that serves as the robot's brain. Regarding its perceptual capabilities, a RGB-D camera is used for user detection in short distance human-robot interactions, while an unidirectional mono microphone is used for recording the verbal activity of the user, in combination with an Automatic-Speech Recognition (ASR) module for speech extraction. Mini has also touch sensors in its belly and shoulders for tactile interactions. Finally, the robot is equipped with an external touch screen for displaying multimedia content and also interacting with users through menus.

Figure 2.2: Mini, a social robot designed for assisting older adults suffering from cognitive impairment.

Mini is equipped with several actuators that provide the expressiveness capabilities of the robot. Although it was designed as a tabletop robot, Mini's body has 5 degrees of freedom: two in the neck, one on each shoulder, and one on the waist. These movements are performed using servomotors that can be controlled in position or velocity. Coloured Light-Emitting Diode (LED) are placed in the robot's chest and cheeks, and allow Mini to express different internal states. The chest LED simulates the heart of the robot, and its intensity, heart rate, and colour can be controlled. Mini also has a LED array in the mouth, synchronized with the audio output of the robot so it works as a VU meter. In the robot's face, two screens are used to represent the robot's eyes. These screens can convey different gazes by displaying a series of predefined GIFs. Finally, a speaker located in its chest endows Mini with the ability to emit verbal and non-verbal sounds. The speech of the robot is generated using a Text-To-Speech (TTS) module.

Because Mini was designed to be used as a research platform, some of its hardware capabilities were oversized, in order to allow the extension of its perception or actuation features as required. Due to the high cost of the oversized hardware features, a new low-cost robot was designed based on Mini, with the objective of obtaining a platform that provides the same features and applications that Mini has, but is affordable by a wider segment of the target population. This new robot was named Gero.

2.2.2 Gero

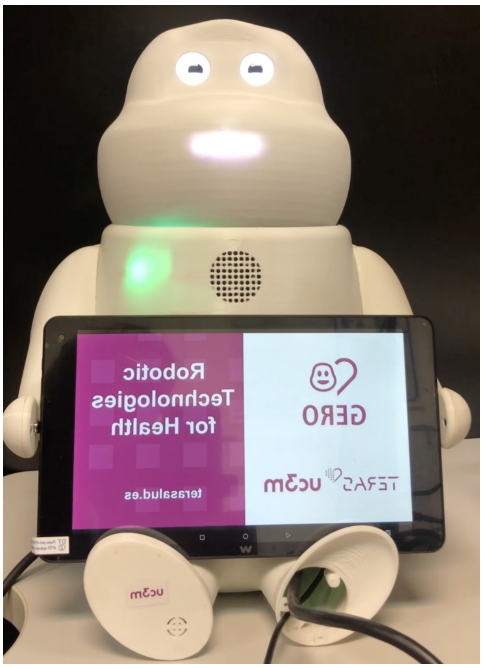


Figure 2.3: Gero, a low-cost social robot designed for assisting older adults suffering from mild cognitive impairment.

Gero is a low-cost social robot designed for assisting older adults that suffer from mild cognitive impairment. It is based on the robot Mini, and was built in collaboration with a company that specializes in healthcare technology-based applications. It offers a subset of all the applications developed for Mini, while its hardware architecture has been resized so it can perform all the required tasks with the lowest cost possible.

Like Mini, Gero is also a 3D-printed tabletop robot, although it switches Mini's soft and squashy appearance for a fully printed shell. It maintains the coloured LED in heart, cheeks and mouth, the speaker located in the chest, the microphone, and the touch sensors, but removes the RGB-D camera installed in Mini. This was decided due to concerns being raised about the possibility of users perceiving a robot with a camera as a threat to their privacy. Gero also has limited mobility, as it cannot move its waist. In Gero, the screen modules used in Mini were replaced by TFT screens, which allow for a manual control of the design of the eyes instead on relying on predefined GIFs. The luminosity of these screens is modulated based on the levels of ambient light, measured with a Light Dependent Resistor.

Although Mini and Gero present some differences regarding its hardware configuration, and the capabilities that each platform has, both share the same modular software architecture, which has been entirely developed by researchers of the Social Robotics Lab. This architecture will be introduced in the next section.

2.3 The control architecture

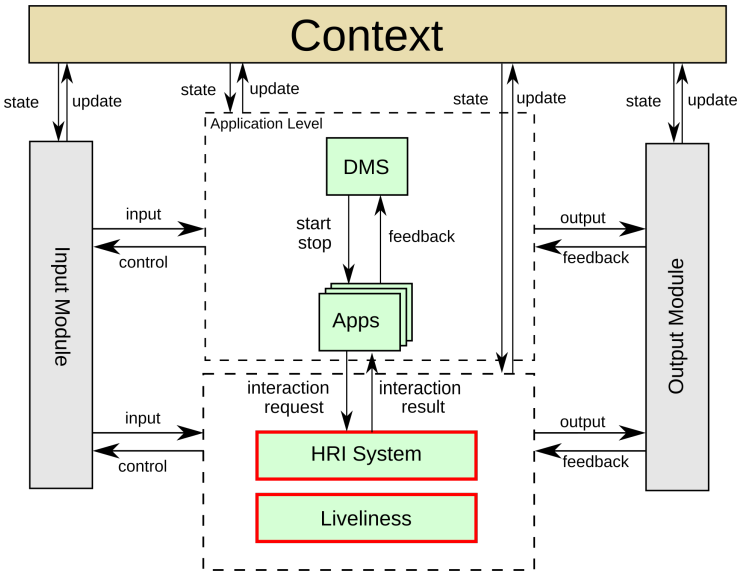


Figure 2.4: Software architecture for the robots Mini and Gero. Green boxes are common to the robotic platforms and grey boxes are platform-dependent.

The software architecture developed in the Social Robotics Lab [28] has been designed with a key concept in mind: modularity. This simplifies the process of upgrading, removing, or adding new modules and features to the system. An overview of the software architecture can be seen in Figure 2.4. In this diagram, the following blocks are present:

- **DMS:** the Decision Making System module, in charge of selecting the behaviour of the robot at any given time.
- **Apps:** the applications that represent the different tasks that the robot can perform. For example, one app can allow the robot to read the news to the user, while another can implement a game between the robot and the user. Applications are started and stopped by the DMS, and they return feedback indicating the progress of their task.

- **HRI System:** module of the architecture in charge of controlling all interactions between the robot and the user. It is in charge of processing all information captured by the sensors, decide the most appropriate way to advance the current dialogue (based on the needs of the applications), and display all the communicative actions that have to be performed. This module receives from the applications requests to start interactions, and sends back to them the result of these dialogues.

- **Liveliness:** this module generates behaviours designed to endow the robot with a liveliness appearance.

- **Input modules:** software modules that control the robot's sensors. They relay the information perceived by the sensors to the rest of the architecture. The rest of modules can send commands to the input modules in order to modify their configuration. This modules are platform-dependent.

- **Output modules:** software modules that control the robot's actuators. They receive commands detailing the actions that have to be performed, and can return feedback about the execution of said actions. This modules are platform-dependent.

- **Context:** module that represents the robot's memory and details its state at any given time. All other blocks can retrieve this state or update it.

The work presented in this dissertation has been integrated in the blocks highlighted with a red frame. For any application that requires that the robot interacts with users, the architecture divides the control over the robot's decision processes in two levels: the Application Level and the HRI System. The Application level decides what the robot has to do at any given time and controls the correct execution of the robot's behaviours, based on task-related information (knowledge required for advancing the tasks the robot performs). Whenever one of the applications has to establish a dialogue with the user, the HRI system takes command and controls the flow of the interaction to achieve the communicative goal requested by the application. This process involves the use of interaction-related information. The applications and the DMS can communicate directly with the input and output modules of the robot for tasks that do not involve an interaction with the user. An example of this could be an application that sends commands to the robot's navigation system. However, all the information that has to be retrieved through an interaction (for example, asking the user his/her name) and all the communicative actions of the robot (for example, the robot greeting the user) have to be managed by the HRI system.

2.3.1 Application Level

As stated before, in the Application level is where the decisions about the robot's behaviour are made. This level is conformed by the Decision Making System, or DMS, and the applications. In our architecture, the DMS serves as the supreme control module, and directs the general behaviour of the robot, while the applications represent all the tasks that the robot is able to perform. At any given time, the DMS can request the activation or deactivation of any application, based on the internal state of the robot, the knowledge it has about the user and his/her preferences, and the information extracted from the environment, including verbal and non-verbal communication with the user.

2.3.1.1 Decision Making System

A Decision Making System, when applied to robotics, is the module that endows a robot with the ability to make autonomous decisions. The DMS uses the available knowledge about the internal state of the robot and the current state of the environment to select the next action of the robot. In our platforms, this is the highest layer of the robot's software architecture, and has control over all the other systems. Two different approaches were developed: (i) a state machine is used to model the tasks that the robot can perform [28]; and (ii) a bio-inspired approach that uses needs and motivations to drive the robot's behaviour [29]. Thanks to the modular design of the architecture and the definition of standard interfaces among modules, replacing one approach with the other is a simple task.

Although there is a significant difference in the process that each of the two DMS integrated in our robotic platforms follows in order to select the next action of the robot, their interaction with the rest of the software architecture is fairly similar. The actions selected by the DMS correspond to the different applications integrated in the robot. The DMS requests the activation of one of these applications, and then waits until the application completes its task, or until changes in the external or internal state of the robot require the interruption of the current task and the execution of a new one.

2.3.1.2 Applications

The applications are the modules that provide the functionalities, or tasks, of the robot. They follow a common template that provides a standard interface between the application and the rest of the architecture. This eases the integration of new applications in the robot. Applications start always in an idle state, and can be activated, deactivated, and paused by the DMS. Also, at given intervals,

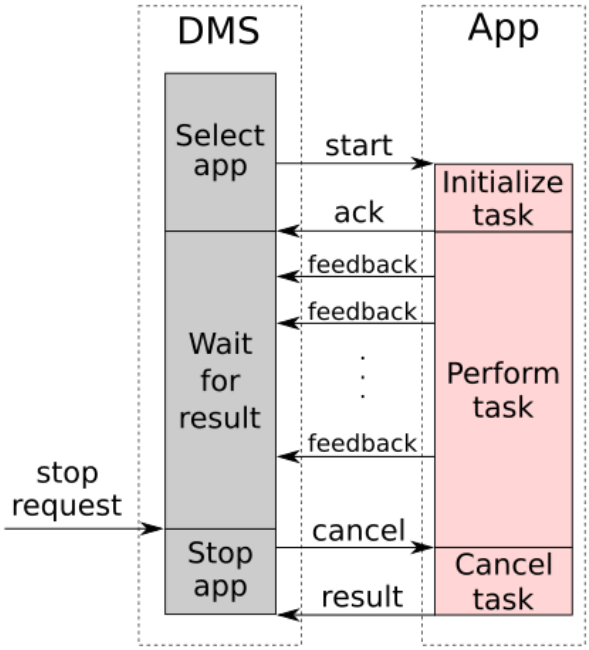


Figure 2.5: Example of the life cycle of one of the robot's applications. The app receives an activation request, and performs its task until a deactivation request is received in the DMS.

the applications can give feedback to the DMS about the state of the task. When developing a new application, the roboticist only has to create the loop that controls the execution of the task, knowing the structure that the activation request is going to have, and also the structure that the feedback expected by the DMS requires. An example of the control loop integrated in the apps can be seen in Figure 2.5

Mini's and Gero's applications have been designed around the idea of a robot that acts as a personal companion for a single senior, and is installed in his/her home, or in daycare centres. This led to considering exclusively one-to-one interactions with humans during the development process. The applications developed for these robots can be divided in two general categories: cognitive stimulation and entertainment. In relation with cognitive stimulation applications, Mini and Gero were developed to help older adults with mild cognitive impairment by presenting them with cognitive stimulation exercises and therapies. A list of exercises derived from applications used by physicians in stimulation therapies were integrated in the architecture.

There is two different ways for the robot to perform these exercises:

- Individual exercises can be completed as part of the robot's entertainment applications. They can be started by the robot, or requested by the user.
- The user's physician can prescribe a customized therapy composed of a combination of these exercises. The robot offers the user to complete the therapy, based on the physician's prescription.

Regarding the applications that are oriented to providing entertainment to the user, these include [30, 31]:

- **Games:** We have tested the implementation of several popular games where we replace one of the players with the robot. In the current version of the robot's architecture, there are two games available. The first one is Bingo, a game in which players have cards with random numbers, and have to scratch them as they are drawn by a caller. In our implementation, the robot plays the role of both caller and player at the same time. The second one is Tangram, a game where the user has to create different shapes proposed by the robot using a set of pieces.
- **Information:** We implemented this category of apps as a tool for the user to keep in contact with the outside world. The robot can download the latest news and read them to the user, while displaying relevant images in the tablet. The user has the choice of selecting the type of content he/she wants to hear about (current news, sports, international, etc...). The robot can also provide a weather report for the current location of the user.
- **Multimedia player:** The robot can display multimedia content using the touch screen in order to entertain the user. The options available include movies, songs or audiobooks, but also personal photos showing the user's family, his/her home town, relevant moments in his/her life, etc...
- **Jokes and Sayings:** This application compiles a collection of jokes and popular sayings that the robot can say. While the sayings are completely random, the user can select the theme for the jokes (animals, robots, etc...), or just ask for a random one.

2.3.2 Context

In the presented software architecture, the context is used as the robot's memory, both short and long-term. Short-term memory is used to store non-persistent data that is generated during the operation of the robot, while long term memory stores persistent data about the robot, the users,

and the environment. Developers can add new knowledge to the context by hand, or it can be downloaded from an online database. Besides serving as a knowledge storage unit, the modules of the software architecture can share in the context any information considered relevant for the whole system. This makes the context one of the main methods for sharing asynchronous information among applications, the DMS, and the rest of the architecture.

The knowledge stored in the context can be categorized in the following categories:

- **User data:** Personal information about a user (e.g., his/her name or date of birth).
- **Agenda:** Information regarding the temporal scheduling of specific robot actions. For example, it could be programmed that the cognitive stimulation therapy has to be started at 18:00 every day.
- **External state:** Contextual information about a user regarding his/her capabilities (e.g., his/her level of proactivity, if he/she has any disability that can difficult interactions, like having hearing problems).
- **Internal state:** Internal information about the robot. (e.g., its affect state, location...)
- **Multimedia:** Details about the content that can be displayed through the robot's touch screen (in particular, their URL).

2.3.3 HRI system

The HRI system is the level of the software architecture in charge of managing all communications with the user. This is the module where this thesis is framed. This system receives interaction requests coming from the Application level, sets up the appropriate dialogue based on the communicative goal that the interaction request defines, uses the relevant information coming from the perception modules of the robot to advance the dialogue, and conveys information to the user through the expression modules. This part of the architecture is divided in three modules that control all the perceptual (Perception Manager) and expressiveness (Expression Manager) capabilities of the robot, when used for interaction tasks, and also manage the flow of the dialogue and advance the interaction based on the information captured by the robot's sensors (HRI Manager). Figure 2.6 shows an overview of the HRI system and the exchange of information with the rest of the architecture. The blocks highlighted with a red frame are the ones that fall under this thesis' scope.

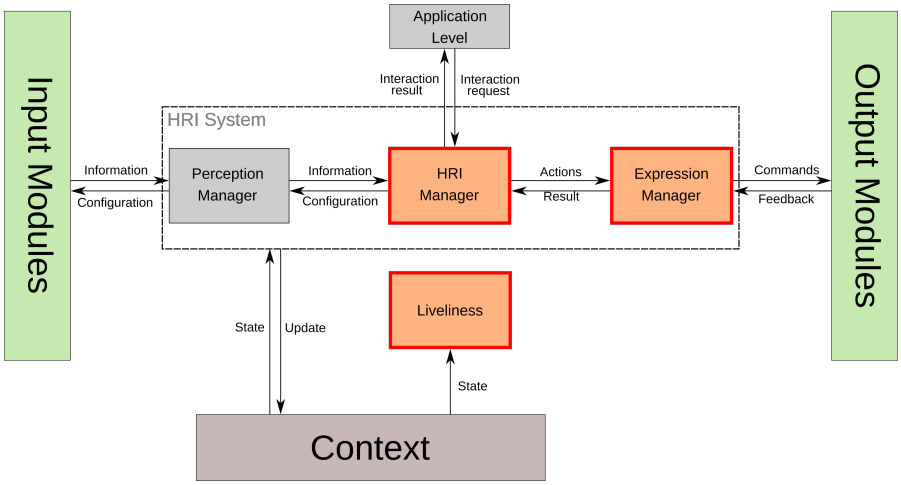


Figure 2.6: Overview of the HRI System and the Liveliness module inside the software architecture.

The input information coming from the sensors is processed by the Perception Manager using three levels of abstraction. In the lowest level, the information is simply formatted into a standardised message as it is received, processing each input modality independently. In the middle layer, a temporal aggregation of all the information retrieved inside a time window is performed, but without establishing any relation between the information coming from different sources. Finally, in the highest level of abstraction, the multimodal information is fused in order to infer complex knowledge. For example, if a gesture detection module recognizes a greeting gesture, and a face detection module recognizes the face of an user *A*, the system could infer the following information: *"User A is greeting the robot"*. So far, only the lower and middle levels of abstraction have been integrated in the robot, while the high level has been conceptually designed, but has not been implemented yet.

At the core of the HRI system, the HRI Manager executes the dialogues required by the applications and the DMS. These dialogues are built as a combination of basic units that allow to convey information to the user, retrieve information from him/her, or respond to any unexpected input provided proactively by said user. The HRI System is equipped with a series of mechanisms for managing any unexpected situation during the interaction (including errors in the perception of the environment, or unexpected behaviours from the user) in a way that assures that the communication does not break, and that the communicative goal can be achieved. With this approach, the applications can model their own complex dialogues as a sequence of communicative goals that have to be fulfilled, while the HRI System controls the interactions required for achieving each goal, and also manages other tasks related with dialogue management (for example, error handling).

Finally, the HRI system also has the control over the expressiveness capabilities of the robot, through the Expression Manager. This module uses a library of predefined gestures that have been handcrafted by the developers. Based on the needs of the interaction, the appropriate expression is performed in order to convey a message. There is the possibility of selecting individual actions for each communication modality (a sentence that has to be uttered, a particular motion for one of the arms, etc...) and create a multimodal expression by combining them. The expressiveness of the robot can be adapted to convey different internal states, through various modulation strategies. Besides executing these expressions, the HRI system also ensures that there are no conflicts when trying to use the different modalities (for example, more than one expression trying to use the same communication channel at the same time).

Both the modelling and management of interactions and the design of multimodal expressions for a social robot are part of the work developed in this thesis. Chapter 3 of this dissertation will focus on the dialogue management aspects of the HRI system, while Chapter 4 will present an in-depth description of the expressiveness management features of the proposed system.

2.3.4 Liveliness

The Liveliness module is in charge of generating random behaviours that do not have a specific communication intention, but instead complement the actions of the rest of the architecture in order to make the user perceive the robot as a living being. The behaviours generated by the Liveliness module can be adapted online so they represent different inner states of the robot.

Chapter 5 of this manuscript presents the two approaches that have been developed for endowing the robot with a liveliness appearance. The first method discussed is the implementation of the Liveliness module integrated in the robot's software architecture is presented in detail. This approach is based on a sinusoidal signal that represents the pulse of the robot. This concept takes inspiration from the human heart rate, and its variations depending on a person's internal state. The pulse of the robot is defined by its amplitude and its frequency. While the amplitude of the signal is only going to be used to modify the actions generated by the Liveliness module, the frequency can also alter the time it takes for new behaviours to be generated. Depending on the robot internal state, the amplitude and frequency of the signal can be altered, so the appearance of the robot changes in order to convey said state. At the moment, the only internal states that are considered are the emotions of the robot. For example, happiness would be translated into an increase in both amplitude and frequency, while sadness would be translated into a decrease of both parameters. The

second approach presented consists on a method for enhancing the robot's verbal communication by pairing its speech with an appropriate sequence of predefined gestures. The selection of these gestures is based not only on the content of the speech, but also on the intention that the uttered sentence has. In this case, the concept of intention is defined as the end goal that the speaker expects to achieve with that sentence. For example, the utterance "How old are you?" has the intention of obtaining personal information from the other speaker. The proposed method combines several machine learning models in order to first extract the intention of the sentence from the verbal content of the speech, and then generate the required sequence of gestures.

2.3.5 Input Modules

Being able to perceive the environment is a key feature for a social robot, as the information extracted from the world is going to affect the decision making process of the robot (as shown in Section 2.3.1.1), and also will change how the interactions between the robot and the user will be conducted. The software architecture presented in this section is equipped with a collection of input modules that can retrieve multimodal information from the environment and the users interacting with the robot. This perceptual information can be classified in four categories, according to its type:

- **Visual information:** As stated before, Mini is equipped with a RGB-D camera. Using this sensor, Mini can detect the face of the users in front of it using RGB images, and upper-body information from depth data. The depth data can also be used to identify gestures performed by the users.

- **Auditory information:** The speech of the user is recorded using a microphone, and then sent to an ASR module, which first extracts a list of words from the speech and then extracts the lexical meaning of this sequence of words using handcrafted grammars.

- **Tactile information:** The touch sensors placed in the robot's body notify if the robot is being touched or not, and where. This is done based on which sensor is detecting the touch. Also, Gamboa-Montero et al. [32] proposed a new tactile perception architecture that uses contact microphones to detect not only if the user touches the robot, but also locate where the contact was made, and classify the type of contact (stroke, hit, etc...).

- **Screen-based information:** All of the robotic platforms presented in Section 2.2 can use a touch screen to display menus for interacting with the user.

2.3.6 Output Modules

While being able to extract information from the environment is crucial for social robots, the range of actions that said robots can perform with their output interfaces will also play an important role on how humans perceive them. Endowing the robot with the ability to convey both verbal and non-verbal messages is essential for human-like communication. Mini and Gero can act over the environment using one of the following types of actions: body movements, gaze, coloured LED patterns, auditory communication, and multimedia content display.

- **Body motions:** both Mini and Gero have motors that can be controlled independently, using position, speed, and acceleration commands.
- **Gaze:** different strategies are used for Gero and Mini. Mini uses a set of fixed gazes that can convey different affect states, motions and positions of the eye, as well different blinking speeds. On the other hand, Gero's eyes allow for a dynamic generation of the gaze. The module in charge of controlling Gero's gaze can modify the shape, size, colour, and position of each of the elements that conform the eye, and combine them in order to dynamically generate different expressions.
- **LED patterns:** RGB LED located in the chest and cheeks of the robots can be used to convey different internal states with a change in colour, brightness, or fading in and out. While the chest LED can use all of these features, the ones placed in the cheeks have been limited to use only the colour red, as multicoloured cheeks could be perceived as strange by the users.
- **Voice-based communication:** the robotic platforms presented in this chapter can use either verbal communication, thanks to a TTS module that generates utterances from text strings, or also use non verbal sounds like yawns or laughs, in order to enrich the robot's expressiveness.
- **Multimedia content:** the touch screen of the robot can be used to display multimedia content, like images, videos, audios...

2.4 Summary

In this chapter, the robots used during the development of this work have been introduced, along with their software architecture. The chapter started with the introduction of the research group where the research for this thesis was conducted: the Social Robotics Group. The most important robotic platforms developed inside this group were presented, highlighting the two robots in which

the proposed system has been integrated and tested: the robots Mini and Gero. An overview of the hardware for both platforms was given. Finally, the software architecture for Mini and Gero was presented in depth, with a complete description of how it operates. The work developed in this dissertation is connected to three main blocks in the software architecture: the dialogue manager and expressiveness manager that conform the HRI system (along with the Perception Manager), and the Liveliness module. These three blocks will be presented in depth in Chapters 3, 4, and 5 of this manuscript, respectively.

CHAPTER 3

Multimodal Dialogue Management

3.1 Introduction

In Human-Computer Interaction, a dialogue model can been defined as *"a model that represents important information about the static and dynamic structure of the conversation"* [33]. Mark Green proposed a similar definition in [34]: *"an abstract model that is used to describe the structure between a user and an interactive computer system"*. Although both definitions are proposed in relation to the field of HCI, they can be applied to describe the process of modelling interactions between a human and a robot. The dialogue model has to maintain the state of the interaction at every moment, the context in which the dialogue is taking place, and provide strategies for selecting the most appropriate action of the system.

No matter what specific model is used for describing the robot's interactions, there is a need for a system that controls the correct execution of the dialogue. This is called the Dialogue System. As a general definition, a dialogue system is a module designed to control interactions with a human. It has to manage all the aspects involved in communication: extracting and processing input information, maintaining the state of the dialogue, selecting the most appropriate response of the system, and handling the execution of said response. From an application standpoint, dialogue systems can be grouped in two different classes [35]: task-oriented, and chat-oriented.

Task-oriented dialogue systems [36, 37, 38] are designed with the goal of helping the user completing a specific task. The system will manage the flow of the dialogue in the direction that leads towards the completion of an objective, and usually will not be able to manage conversation topics

outside the task's domain. Examples of these systems include online help support systems, or booking assistants at an airline or an hotel, for example. Opposite to task-oriented systems, chat-oriented dialogue systems [39, 40] aim to to provide a general conversation to the user, without a specific goal to achieve or a task to complete. Some task-oriented dialogue systems include a chat-oriented module to manage conversations outside the domain of the task they were built for (or as a fallback mechanism) [41]. The most common example of this type of dialogue systems are what is known as chatbots. [42, 43].

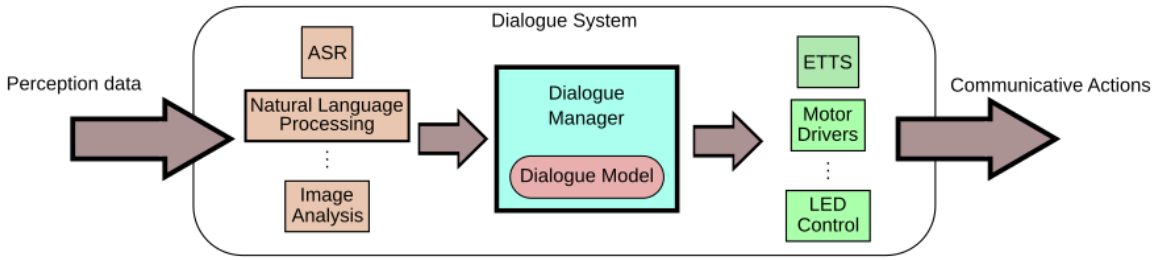


Figure 3.1: Example of the relationship between a Dialogue System, a Dialogue Manager, and a Dialogue Model.

Regardless of the application they were built for, or the channels that can be used to retrieve and communicate information, many dialogue systems have the same module at its core: a Dialogue Manager [44, 45]. According to Lison [13], a dialogue manager is the module that is in charge of maintaining the representation of the current dialogue state (which includes the knowledge that the system has regarding the interaction, the dialogue history, the external context and the communicative goal) and deciding which actions the system has to perform based on the state of the dialogue. Figure 3.1 shows an example of the relationship between a Dialogue System, a Dialogue Manager, and a Dialogue Model. If we consider the definition proposed by Traum et al. [46], dialogue management groups the following four functionalities inside a dialogue system:

1. Update the context of the dialogue based on the information provided by other communicative agents (either human users or other computer systems).
2. Provide context-dependent expectations for how input signals have to be interpreted as communicative behaviours.
3. Interfacing with domain processing modules to coordinate dialogue and non-dialogue behaviour and reasoning.
4. Select the actions of the system and when to express them.

In order to apply the concepts presented in this section to dialogue management in human-robot interactions, the following problems have to be tackled. First, it is necessary a model that represents the interaction and captures all the particularities of the dialogue, the Dialogue Model. Second, it is necessary to develop the software infrastructure that can handle that representation of the interaction, the Dialogue System, and in particular the Dialogue Manager. Figure 3.2 shows the software architecture that was introduced in Chapter 2. The red box indicates the module of the architecture where the work presented in this chapter will be integrated.

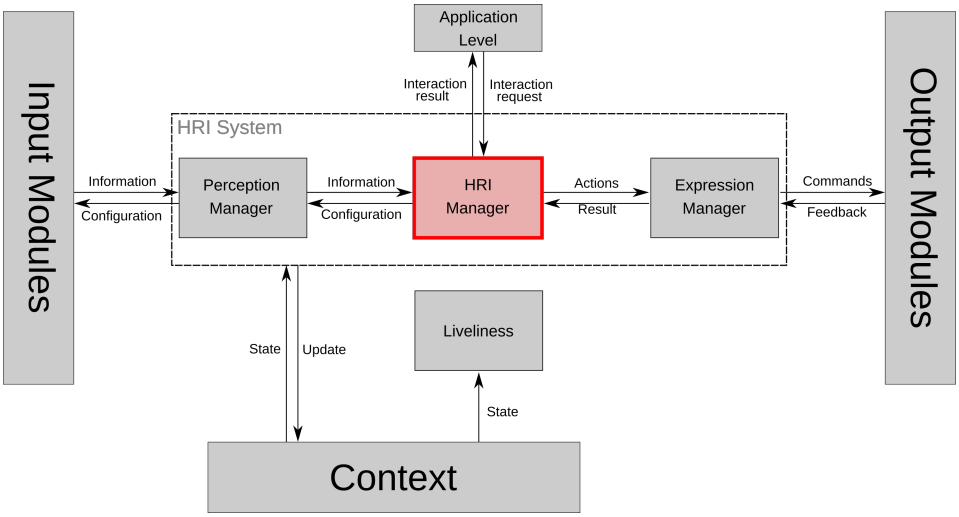


Figure 3.2: Diagram representing the software architecture described in Chapter 2. The work developed in this chapter of the dissertation focuses on the module of the architecture highlighted in red.

3.1.1 Objectives

In relation with the first problem of dialogue modelling, this thesis proposes a modular approach where dialogues are represented as a combination of basic interaction units, called Communicative Acts, or CAs. These CAs are highly parametrizable so they can be used in multiple domains, and can be executed on its own, or combined with other CAs. This approach to dialogue modelling allows to integrate all aspects of interactions that are task-independent in the CAs, leaving the control of the dialogue flow, which usually requires task-related information, to the applications of the robot. This way, developers can use the CAs as a tool to create all the interactions that their applications might need, without having to worry about managing all low-level interaction tasks. Regarding the implementation of the Dialogue System that can handle this dialogue model, this is a very complex task that involves multiple modules. The dialogue system integrated in the software architecture

presented in Section 2 is the HRI System, which at is core has one key component: the HRI Manager. This is the dialogue manager that controls the proper configuration and execution of the CAs.

The final objective of the work developed in this area is directed to endowing a robot with the required abilities for conducting an interaction that is deemed as satisfactory by a human observer. In order to achieve this, the main goal that has to be achieved is **the design and development of a dialogue manager that allows to control multimodal interactions between a human and a robot using a combination of basic, parametrizable dialogue units**. On top of implementing and managing this type of dialogues, this dialogue manager has to also be able to control the interaction-related uncertainties that might arise during any dialogue.

This objectives have a high complexity, and include multiple aspects of both dialogue modelling and management. For each main goal, a series of subgoals were proposed:

1. The dialogue model has to allow the usage of multiple communication channels, both as inputs and outputs. The selection of the input and output channels has to be independent, and any possible combination of channels has to be allowed.

2. The creation of complex dialogue structures has to be achieved through an appropriate combination of more simple dialogue units. Thus, a modular approach to dialogue modelling has to be followed.

3. The dialogues built with the proposed architecture have to abide by the temporal restrictions that apply to regular human-human interactions. This means that the response of the system to an action performed by a human observer has to be delivered in a time that feels appropriate to said observer.

4. The proposed dialogue manager has to overcome communication problems related to errors in perception with an appropriate strategy that avoids a communication breakdown. The system cannot stop an interaction with an user due to partial information being provided by the perception modules.

5. The dialogue manage has to control user-related unexpected situations. This includes changes in initiative during the dialogue, and also a sudden disengagement by the user.

While the first two subobjectives are related to the modelling of dialogues, the last three are connected to the management aspects of the main objective.

3.1.2 Overview of the Chapter

An overview of the contents of each section is presented next:

- Section 3.2: In this section. a review of the state of the art on dialogue management is presented. First, the section gives an introduction to the different approaches that can be followed for developing dialogue managers, and then a series of relevant works from each approach are reviewed. In order to highlight the contributions made with this thesis, a comparison between the proposed dialogue manager and the solutions reviewed is conducted, stating the similarities and the differences.

- Section 3.3: This section focuses on the theoretical aspects of dialogue modelling, and presents a series of concepts extracted from the field of linguistics that have been used for developing the dialogue model proposed in this work. Here, the concept of *Communicative Act* is introduced as the basic unit that will be used to build interactions in this HRI architecture.

- Section 3.5: The dialogue manager developed for this thesis is presented: the HRI Manager, along with all the technical aspects of its implementation. Next, a review of all the features offered by this dialogue manager is conducted. Finally, all the CAs developed for the robotic platforms in which the system has been integrated are presented.

- Section 3.6: This section presents the evaluations conducted for testing the performance of the proposed approach. In this evaluation, the robotic platforms interact with real users in the context of their target application (robots that provide entertainment and cognitive stimulation exercises). A series of objective tests are presented to showcase the system's performance, and a case of use is described to explain how the different features of the proposed dialogue manager are used in a real task.

- Section 3.7: This section compiles the conclusions extracted from the work developed on dialogue modelling and dialogue management. The main contributions of this section of the thesis are highlighted, and a review of the goals that were set at the beginning of this research is conducted. Finally, a series of possible future lines of work in the area of dialogue management are presented.

3.2 State of the Art

This section presents a review of a relevant collection of works in the field of dialogue management. The main objective is to give the reader a comprehensive idea of the different approaches that can be

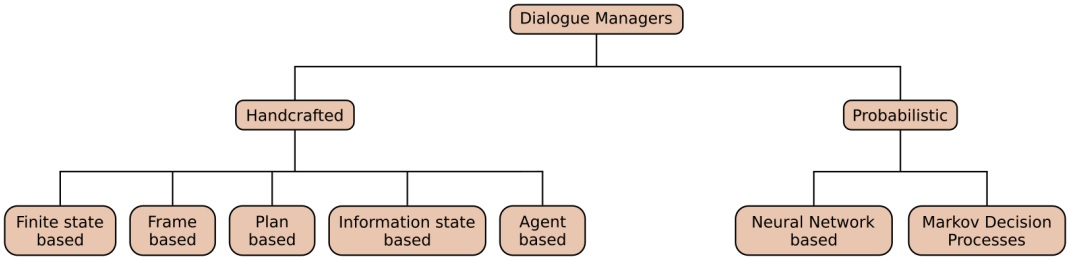


Figure 3.3: Classification of dialogue managers proposed by Keyvanpour et al. [3].

followed to model and manage dialogues, and the main differences between the proposed approach and the rest of the works reviewed in this section. Although the dialogue model and manager proposed in this dissertation were developed to be integrated in a social robot, the review will also include approaches designed for other communicative agents.

3.2.1 Approaches to dialogue management

Depending on the approach used to model the dialogue, there are different taxonomies that can be used to categorize dialogue managers. In 2006, Trung [47] proposed the following categories: (i) finite-state and frame-based dialogue managers, (ii) Information state-based and probabilistic-based approaches, (iii) plan-based approaches, and (iv) collaborative-agent approaches. In recent years, with the advances in the field of machine learning, new dialogue managers and end-to-end dialogue systems have been developed using neural-based approaches. These managers are left outside of Trung's categorization, and thus, require the modification of traditional classifications. In 2020, Keyvanpour et al. [3] proposed a new taxonomy that, while maintaining some of the categories used by Trung, also included the new machine learning-based methods. In their approach, dialogue managers are first categorized in two global classes: hand-crafted and probabilistic. In handcrafted approaches, the developers of the system define the rules that will be used to decide which action the system should perform at every turn. This category includes all the dialogue manager types proposed by Trung. Probabilistic approaches rely on statistics and machine learning techniques for managing interactions. They have the advantage of being more flexible than handcrafted systems, as they can keep learning during interactions and adapt to new situations, but, on the other hand, they require large datasets for training, and its computation can be complex. This type of dialogue managers can either be classified as neural network-based, or approaches that describe the dialogue as a Markov Decision Process. Figure 3.3 depicts an overview of the dialogue manager classification proposed by Keyvanpour et al.

In this section, following the Keyvanpour et al.'s classification, the most relevant works will be presented and analysed in depth. Then, all the approaches will be compared according to a series of characteristics that have been defined explicitly for this research: (i) multimodality, (ii) scalability and re-usability, and (iii) expressiveness.

3.2.1.1 Finite state-based approaches

Finite state-based dialogue managers model interactions using a state transition network. In this case, the states represent the different actions of the system, while the actions of the other communicative agents (humans or other systems) activate the transition between states. Dialogues are handcrafted by developers before hand, and the actions of the system are fixed. On one hand, this approach provides a straightforward method for encoding interactions, but on the other hand, these interactions tend to be deemed as unnatural, as they are rigid regarding which user answers can be accepted, and how the dialogue flow has to advance.

In 2010, Peltason and Wrede [48, 49] presented the PaMini framework, a dialogue modelling approach in which dialogues are designed by combining generic interaction patterns. This framework also includes a novel task-state protocol that connects the dialogue system through an asynchronous event bus with the back-end modules of the robot. This back-end includes all the task servers that provide domain-specific knowledge, but also modules that provide specific perceptual and expressiveness capabilities. In this approach, the modules of the architecture can request the execution of *tasks*. The information required for this execution is stored in the task's *specification*, which can be updated at any moment. The task-state protocol connects the dialogue system with the clients and servers that manage the *tasks*. Under this approach, interactions are modelled using Interaction Patterns, defined as "a sequence of human dialogue acts, robot dialogue acts and system actions" [49]. They represent sub-dialogues that can be reused in different domains. When the developers need to create an interaction, they use a Java API to generate the dialogue acts that will be required, select the appropriate Interaction Patterns related to each dialogue act, and parametrise them with task-related information. Multiple sub-dialogues can be run at the same time, and they can be stopped at any time. While they are self-contained, these sub-dialogues can also be interleaved to create more complex behaviours. The system manages a context with information shared by all the patterns, and they can modify said information at any given time. The API also allows to define different parameters of the interaction, like the strategies for selecting patterns, interleave them, asking the user for a clarification, and for opening the interaction, among others.

3.2.1.2 Frame-based approaches

Frame-based dialogue managers, also known as slot filling methods, were developed as an extension of finite state-based approaches with the objective of overcoming some of the limitations of these approaches. In frame-based dialogues, the system looks to collect a set of information by filling multiple slots, which are grouped in frames. According to Keyvanpour et al. [3], the advantages of this approach are a relatively higher flexibility and a more natural dialogue than the finite state-based, as they allow the user to provide information to fill multiple slots at the same time. But this approach shares some of the disadvantages of the finite state-approach: it is not well suited for complex problems, and also limits what the user can communicate to the system.

In 2000, Souvignier et al. [50] presented in their work a collection of tools used in spoken dialogue systems through the description of their system. Being a Spoken Dialogue System, the perception architecture is based on a combination of an Automatic Speech Recognition module and a Natural Language Understanding module, while the output of the dialogue manager is sent to a text-to-speech module. In this approach, the dialogue manager follows the slot-filling approach. After every user input, the dialogue manager assigns the information extracted from the user's speech to fill as many information slots as possible. Then, the system queries the database searching for a valid response. If multiple responses are found, the system asks for more information in order to fill more slots. In this case, the question would be aimed at filling the slot with the highest disambiguation potential. The system includes the recent slot values stored in the following dialogue turn so the user can request a correction if needed. Under this approach, slots, questions and verification strategies are defined in a declarative way, while the mechanism for filling slots is common to all frames, and stored in the dialogue manager. Thus, the task-specific aspects of dialogue are handcrafted by developers, while the general, task-independent aspects are handled internally by the system. The developers can also specify conditions for emptying the value in the slots. The proposed approach also keeps track of the dialogue history so it can be used in later turns, if the user makes an implicit reference to it. The history is also used to ensure consistency between the values stored in the slots, and that the user did not provide an invalid value (one that does not match any entry from the database, or that corrects a slot value that cannot be altered).

More recently, in 2015, Alonso et al. [51] proposed a multimodal dialogue system for social robots. In this architecture, the information coming from the perception module is sent to a multimodal fusion node that aggregates this data in packages of semantic values. The aggregated data is sent to the dialogue manager, which advances the current dialogue based on this input data. If the dialogue manager decides that an action has to be executed (an action that could imply the use

of multiple modes of interaction), then it is sent to the multimodal fission node, which sends the necessary instructions to all the output modules involved in said action.

In their approach, Alonso et. al used Iwaki, an agent-based dialogue manager for flexible turn-taking in mixed-initiative dialogues, similar to the COLLAGEN manager [52]. The interactions are modelled using a tree-based structure. Two years later, Wessel et al. [53] presented OntoVPA, an ontology-based multimodal dialogue manager for Virtual Personal Assistants. This system is a declarative, knowledge-based system that combines generic, dialogue-specific ontologies with a set of dialogue management rules that implement the dialogue system's core capabilities. The ontologies are used for dialogue and domain representation, as a collection of classes and relation types. On the other hand, the dialogue management rules are in charge of controlling the dynamic aspects of the dialogue.

In OntoVPA, ontologies can be divided in *domain ontologies* and *dialogue ontologies*. The first type contain all the domain-related information structured in classes and subclasses (for example *restaurant* might be a class, while one of its subclasses could be *Italian restaurant*). The domain ontologies also specify the relations between classes. The dialogue ontologies follows a similar class-based structure, but in this case the classes define the dialogue acts of the user (the dialogue acts of the system are generated by the ontology-based rule engine). On the lower level of the system, a set of ontology-based rules are used for generating the actions of the system, as well as the domain workflows and the domain-specific application logic. Tasks in this framework are defined using a frame-based approach, and the ontology-based rules are in charge of how the information provided by the user is used to fill the slots, retrieving missing information, and in general conducting the flow of the interaction. Also, the rules also include conflict-resolution strategies that might be necessary for solving all the uncertainties related to spoken interactions.

Wahde [54] presented DAISY in 2019, a dialogue system for virtual agents based on the use of dialogue building blocks that represent situations that are common to multiple dialogues. While the author argues the proposed system cannot be categorized as either state-based or frame-based, for the purposes of the analysis, it will be considered an example of the latter type, as it maintains multiple objects that try to match the inputs of the user to a series of templates, instead of just keeping an state active and transitions that are triggered based on these inputs. In this system, the agent can handle dialogues through a combination of these dialogue building blocks, and use a sequence of basic actions involving cognitive processing to manage the information required to solve a given task. The dialogue building blocks are

dialogue manager independent. They model small common interactions present in dialogues (e.g., requests of information, branching, error handling, or dialogue flow control...), can manage context-dependent information, and can communicate with the memory of the system to select the appropriate answer to a given request, and to carry out the necessary deliberations to solve this request.

3.2.1.3 Information state-based approaches

Information state-based dialogue managers use probabilistic techniques as a way to overcome the limitations in both finite-state and frame-based approaches. These type of managers usually include the following components: (i) a description of the informational components of the system (like the participants in the interaction, or the knowledge they share, among others), (ii) formal representations for those components, (iii) the dialogue moves that will lead to an update of the information state, (iv) the update rules that will control the update of the information state, and (v) the update strategy that the system will use to select which update rule to apply at any moment.

In 2012, Morbini et al. presented FLoReS [55], a combination of information state-based and plan-based dialogue manager that makes use of forward inference, local dialogue structure, and plan operators that represent the structure of sub-dialogues. This manager consists of the following elements: (i) an information state that keeps the history of the interaction and any information that might be relevant; (ii) a set of inference rules for adding new knowledge to the state, based on conditional relationships between the state variables; (iii) an system for handling events that might trigger an update of the state (these events might come from the user, the system, or the environment), and (iv) a set of operators that handle the modelling of the dialogue structure and select the next steps in the interaction. These operators can be reused across different domains, and are modelled as a tree of system and/or user multimodal actions, and resulting states. Each state in the tree is associated with a communicative goal, and provide a reward for reaching said state, which is stored in the information state. If multiple operators can be used in a situation, the goal of the system will be used to decide which operator to use. Operators in this approach can have preconditions, that indicate when the operator can be activated and in which state should start, and effects, which produce modifications in the information state.

The interaction flow in the FLoReS dialogue manager works as follows. When an event is received, it is matched against the different event listeners in order to decide which action or actions should be performed. If these actions lead to a change in the information state, then the inference rules are evaluated continuously until the state is stable. Next, the dialogue manager decides which

operator should be triggered. If the active operator cannot handle the event, the system computes the rewards obtained in two scenarios: ignore the event and trigger an operator that handles system initiative (either the current one or switching to a new one), or handle the event with one of the paused or inactive operators. The option of ignoring or reacting to the user's input allows to create mixed-initiative interactions. The reward for each operator is computed by simulating the possible dialogues that can happen, starting with the current information state, until the simulated dialogue reaches a termination condition. After all rewards have been computed, then the operator with the highest estimated reward is set as active, and the current active operator is paused, if possible, or deactivated.

Three years later, Pierre Lison proposed in 2015 [56] a hybrid approach to dialogue management that relied on the concept of probabilistic rules: structured mappings between logical conditions and associated probability distribution effects. Probabilistic rules represent the transition and utility models of a dialogue POMDP. In this approach, the dialogue state is represented by a Bayesian network, while its variables represent different aspects of the current context. The probabilistic rules are applied on runtime to update the state based on new observations, and to select the next action of the system. These rules are implemented following an *if-else-then* structure, and formally expressed as an ordered list of branches. Each branch is a pair composed by a logical condition (a logical formula over a subset of the dialogue state variables) and a categorical probability distribution over a set of effects that have to be mutually exclusive, and that lead to changes in the rule's output variables. The probability associated to each effect can either be manually defined or estimated empirically. The formalism described for probability rules can be extended to express utility functions, this is, how to select the action with the highest expected utility for every state. This is done through the utility rules. They follow the same structure than the probability rules: an ordered list of pairs composed of a condition and an associated utility table over possible assignments of specific values to decision variables. This utility table connects each possible decision with its utility value. For example, a specific action might have a utility value of 5 if it has been requested, but -5 if it has not. This framework was developed using OpenDial, a domain-independent platform for developing probabilistic spoken dialogue systems. In 2018, Milhorat et al. [57] proposed a conversational dialogue manager for the android ERICA. This dialogue manager is composed of four elements: a question-answering engine, a statement response module, a backchannel module, and a proactive initiator. A top-level controller is in charge of selecting which module should handle the next interaction, depending on the current state of the dialogue.

When an utterance is received from the user, the system sends it to the question-answering engine and the statement-response module. These modules classify the utterance as either a question or a non-question through a dialogue act tagging process. Then each module returns a confidence score based on the dialogue act, and possible keywords and focus phrases identified in the utterance. The system will select the module with the highest confidence score, if its over a threshold, or it will fallback to the backchannel module. The question-answering module uses a handcrafted database of adjacency pars to manage its knowledge. These match user utterances with system responses, so for each utterance coming from the user, the system selects the closest example in the database and executes its corresponding response. If there is no examples that are similar enough, the system falls back to the backchannel module. Statement responses are based on a partial repetition of the user's utterance. The system starts by selecting the focus phrase in the utterance, then extracts all the nouns in this phrase, and finally searches for the combination of a focus noun and a question word using a n-gram language model. If the probability of this combination is above a threshold, then the system action will be the generated question. Otherwise, it will be a partial repetition of the user's statement, using a raising tone to indicate a question. If no focus phrase is found, then it will try to generate a question on the utterance's main predicate, or will fallback to a backchannel response if this also fails. Finally, a proactive initiator is used to respond to different events (pauses in the dialogue of different lengths, or an user approaching/leaving the robot). Strategies for handling these events including asking a question about the most recent topic, pick a new topic close to the current one, or the activation of greeting or farewell dialogue.

A year later, Kiefer et al. [58] presented VOnDA, a framework for implementing dialogue management functionality in dialogue systems. It works not as a complete dialogue manager, but as a implementation layer for building complex reactive systems. VOnDA implements the information state approach to dialogue management as a combination of a rule-based approach and statistical selection. The information state is built using an RDF (Resource Description Framework) store and reasoner, which can assign temporal information to every data chunk stored, obtaining a complete history of all changes that have been effected over an object. This RDF store contains all the knowledge the system has about the hierarchy of dialogue acts, semantic frames, and objects, as well as specifications for the type and properties of data.

Data coming from the sensorial modules of the system or the applications, speech recognition results, or expired timers can effect changes over the information state, which in turn triggers the evaluation of a set of reactive condition-action rules, which can cause more changes, the execution of actions, or the generation of a *proposal* (a block of code kept in storage). If the system reaches a stable

point (no more state changes or proposals generated), then the stored proposals are evaluated to select the best way to continue the dialogue.

3.2.1.4 Plan-based approaches

Plan-based dialogue managers describe any dialogue as a goal that has to be achieved. Inputs to the dialogue are described as *speech acts* [59] that will be used to achieve those goals. The listener in the dialogue is expected to understand the plan and help the speaker to fulfil it. The main advantage of this approach is its scalability, while its disadvantage is that its performance suffers in complex domains, although it performs better than frame-based and finite-state based [3].

In 2000, Jennifer Chu-Carrol [60] presented MIMIC, an adaptative mixed-initiative spoken dialogue system for information queries through phone calls. In this approach, the input is first transformed in a frame-based semantic representation, and then used to generate a response, along with the domain knowledge and previous history. The response generation is a three-step process. First, the system detects cues (based on semantic representation and history, or on domain-specific knowledge) present on the user's utterance, and uses their effect to adapt the initiative for the turn, modelled as a *basic probability assignment* (probability of each speaker holding the initiative). The speaker with the highest degree of support for having the initiative will be considered the initiative holder. Once the initiative has been determined, the system selects the goal that has to be achieved with the response, using information from the cues extracted from the user's utterance. Next, a strategy for achieving the selected goal is planned, accounting for the initiative distribution. Finally, MIMIC uses a template-driven utterance generator to create the system's response.

In 2009, Bohus et al. [61] presented RavenClaw, a plan-based, task-independent dialogue management framework that separates the domain-specific aspects of the dialogue from domain-independent conversational skills. In this framework, the developers design the dialogue flow, while RavenClaw handles low level tasks, like error-handling or turn-taking. The objective is to simplify the development of mixed-initiative systems for complex, task-oriented domains.

The dialogue specifications designed by the developers are executed by the dialogue engine. It is composed of a dialogue stack, which maintains the structure of the dialogue during runtime, and an expectation agenda that stores predictions for the user's actions at any given dialogue turn. The execution of a dialogue can be divided into the execution and input phases. During the former, the system executes the routine for the agent at the top of the stack (initially, this will be the agent at the

top of the dialogue tree), which will perform different actions depending on the agent, including adding new agents from the dialogue tree to the stack. Then, the system removes from the stack any agent that has been completed. The next step is running the error handling routines if necessary. RavenClaw can manage errors related to two common situation in speech-based interactions: non-understandings and misunderstandings. The error handling decision process decides which strategy (if any) should be performed. Individual modules generate a series of strategies, which are then queued by a gating mechanism into the dialogue stack. Finally, a new agent is selected from the stack, based on a series of conditions defined for each agent. If the agent executed in a dialogue turn requires an input from the user, then an input phase is performed. The system starts by composing the *expectation agenda* with the expected input from the user. When an input is received, the information contained in it is used to update the concepts in the agenda (taking into account the different hypothesis for each concept, and also context information). The end of the input fase leads to a new execution phase. User initiative is controlled by defining which *expectations* in the agenda can be updated.

RavenClaw also offers a series of functionalities that will be required in multiple dialogues. This includes mechanisms for handling timeouts, repeating utterances, or stopping and resuming the interaction, among others. These features are modelled as dialogue agencies, and can be specified by the developer during the dialogue task specification creation. The engine will invoke these functionalities when required by the situation. System developers can create new conversational strategies and add them to RavenClaw.

3.2.1.5 Agent-based approaches

In agent-based approaches, dialogues are described as a collaborative process between communicative agents. The objective is that all the agents involved in the process achieve a mutual understanding of the dialogue. The agent-based approach to dialogue management focuses on the motivations behind an interaction, and its intrinsic mechanisms, instead on focusing on the task at hand. The system models the beliefs of the participants in the dialogue and, similar to the plan-based approach, defines a goal that will drive the actions of all the agents. This type of dialogue managers are hard to craft due to their complexity, but they provide a natural dialogue and can handle complex tasks [3].

In 2009, Turunen et al. [62] presented an adaptive architecture for interaction and dialogue management in spoken dialogue applications. This architecture uses dialogue agents, evaluators, and managers to represent interaction strategies and the coordination among them. The agents are in

charge of presenting the interaction techniques, the evaluators select the most appropriate agents for a specific situation, and the managers coordinate the whole system. The dialogue system is divided in three modules that follow the same structure: an input model, a dialogue model, and a presentation model.

The interaction agents implement different interaction strategies, and can be specialized on performing specific tasks (for example, error handling). Each agent has a set of attributes that indicate what task can they perform, and how proficient they are. The managers are the modules that control the interaction tasks. They evaluate how well a strategy suits the current situation, based on the attributes of each agent and the context (dialogue history, current inputs, or user preferences), and select the agent that should handle it. The interaction manager is the one that controls all the others, and is designed by the application developers to suit their needs. The comparison between agents is done by the evaluators, which focus in different aspects of the agents. For example, one can evaluate the output modalities of the agents, while other might focus on how well the agents fit with the discourse history. The evaluation from each evaluator is combined to generate a final score. In the proposed architecture, three different models exist. The dialogue model handles the communication between the user and the system, while the input and presentation models focus on the interaction tasks. An interaction is then defined as a sequence of dialogue units formed by the agents of the dialogue model. This dialogue management approach supports different dialogue strategies (i.e. mixed-initiative, or system-initiative), dialogue control models (state machine, forms...) and reusable dialogue components.

In 2012, Nestorovič [63] proposed an agent-based dialogue agent based on the *Beliefs-Desires-Intentions* architecture. This dialogue agent is composed of five modules: a context module, a history module, a strategy selection module, a core module, and a prompt planner module. The context module keeps information about the state of the dialogue. The history module is in charge of solving contextual references to the dialogue history. The strategy selection module decides which is the most suitable initiative mode for the agent's response in familiar situations. The core module selects the next action of the system based on the current state of the dialogue. Finally, the prompt planner module receives the action selected by the core module and transforms it in a utterance. In the version of the system presented in [63], the last module is not integrated, and the action selected by the core module is sent directly to the TTS module.

3.2.1.6 Neural Network-based approaches

Advances in the field of machine learning, and specifically in the area of deep learning have led to the use of neural networks for dialogue management tasks. These approaches aim to overcome the domain constraint of handcrafted approaches by learning from a dialogue corpus. This solution has the advantage of automatically generating the dialogues, without the need for handcrafting any part of the process. However, it requires large training datasets to train the different modules of the system [3].

In 2016, Su et al. [64] presented a network-based model initially trained with a supervised spoken dialogue dataset, and then improved through a reinforcement learning process in which the system is used to interact with a simulated user. This has the advantage of combining both machine learning strategies without having to modify the architecture, and also the system can operate on a full set of actions, instead of constraining it to minimize the training cost.

In this approach, the system receives a belief state s that encodes the recognized intents from the user and the dialogue history. On the other end, the manager outputs actions that decide the system's semantic reply, which is then sent to a Natural Language Generation module. The dialogue manager is represented with a Policy neural network that predicts dialogue acts and queries, and manages offers made by the system (slots of information the system can mention). The goal of the system is to decide what the intent of the system's response has to be (based on the knowledge about the user) and also to select the slot of information that the system should talk about. The actual values for the slots are decided by a database parser. The query to the database contains the top prediction for each user-informable slot (slots used by the user to constrain the topic) in the dialogue state, while the response is the system's semantic response. In an initial phase, the policy network is trained on corpus data, and then refined through a policy-gradient based reinforcement learning training.

That same year, Cuayáhuitl et al. [65] proposed a deep reinforcement learning approach to dialogue management based on a network of Deep Q-networks, or DQN. In this approach, each DQN represents a conversational skill for a particular sub-dialogue. This approach allows for transitions between all DQN agents in order to provide flexible and unstructured dialogues. Also, when a user response leads to a change of topic, the completion of the sub-dialogue in the new domain triggers a transition back to the previous topic, so the interaction can be resumed.

A year later, Wen et al. [66] presented a network-based, task-oriented dialogue system. Their model follows an end-to-end approach to training, while still being a modular system. In this work, dialogue is modelled as a sequence-to-sequence mapping problem. For each dialogue turn, the model

generates two different representations of the user's input: a distributed representation generated by an intent network (authors tested a convolutional neural network and a long-short term memory network), which can be compared to dialogue act representation, and a probability distribution over the belief state (task-related information), represented by a set of slot-value pairs, and generated by a set of belief trackers. The system has an specialized tracker for each slot in the ontology. Then, the database operator queries the database with the values of the belief state that have the highest probability. The entry returned by the database, the intent representation, and the belief state are transformed and combined by a policy network into a vector that represents the next action of the system. The vector is then sent to a generation network that generates a template as a sequence of tokens, using the language model probabilities. This template is then filled using the information retrieved from the database. All the modules in the model are connected by the policy network.

Also in 2017, Li et al. [67] presented an end-to-end learning framework for task-completion dialogue systems. The proposed framework includes a user simulation for training the dialogue management module. The other elements in the system are a language understanding unit, the dialogue manager, and a back-end database. The components are trained using a reinforcement learning approach. In the dialogue system, a language understanding unit generates a semantic frame with the relevant information in the user's utterance, as well as his/her intention. Both tasks are performed simultaneously by a Long-Short Term Memory network trained using back-propagation. The system generates a concatenated sequence of IOB-format slot and intent tags. The resulting semantic frame is sent to the dialogue manager, which then performs a two-step process: dialogue state tracking and policy learning. During the dialogue state tracking phase, the system starts by querying the database with the information contained in the semantic frame. Then, the dialogue sate is updated based on the user input and the result of the query. Finally, the state tracker prepares a state representation to be used during the policy learning phase. This representation contains the last actions from the user and the system, the information extracted from the database, information about the current dialogue turn, and historic information about the dialogue. Based on this representation, the policy has to decide what the next action of the system should be. The policy is represented using a deep Q-network, and can be optimized through supervised learning or reinforcement learning.

That same year, Eric et al. [68] presented a neural dialogue agent for grounded multi-domain discourse through an attention-based key-value retrieval mechanism over a knowledge base. This makes the system able to learn how to extract relevant information from the knowledge base without requiring an explicit training of belief or intent trackers. The proposed model uses an encoder-decoder architecture augmented with the attention-based retrieval mechanism. The encoder

of the dialogue system is represented by a Long-Short Term Memory network. In a given turn, the system encodes an aggregated dialogue context composed of the system and user utterances in all previous turns. First the sequence of tokens is embedded using an approach that maps each token to a fixed-dimensional vector, and then fed to the encoder to generate a context-sensitive hidden representation. Then, a sequence-to-sequence decoder predicts the system response.

In 2019, Xu et al. [69] presented an end-to-end dialogue system that adds domain knowledge graphs to the dialogue management topic transition. In particular, the application considered in their work is medical diagnostics. The proposed model includes a knowledge-routed deep Q-network that combines a relational refinement branch for encoding the relations between symptoms and diseases, and a knowledge-routed graph branch for making decisions about dialogue topics.

The dialogue system can be divided in three modules: natural language understanding, dialogue manager, and natural language generator. The overall pipeline of the system is as follows. First, the natural language understanding module extracts the intents of the user and the information that will be used to fill a set of slots from the utterance. In this module, a Bi-directional Long-Short Term Memory network tags words in the sentence with BIO labels and extracts the intention simultaneously. Then, information slots are filled based on the context of the dialogue and medical term normalization. The dialogue manager controls the transition between topics based on the state of the context. The interactions with the user are conducted according to a dialogue policy designed to select the action that maximizes the future reward (encouraging making the correct diagnosis with precise symptom request in the minimum number of turns). First, a preliminary action is generated using a deep Q-network with a Multilayer Perceptron, based on the state of the dialogue. Then, in the relational refinement branch, the system can modify individual symptoms or diseases through the aggregation of information regarding other related symptoms or diseases. Parallel to this relational refinement process, the knowledge-routed branch starts by computing the conditional probabilities between diseases and symptoms in both directions (from disease to symptoms and from symptoms to diseases). Based on the candidate diseases and symptoms probabilities, the system generates the knowledge-routed action probabilities. Finally, the preliminary action, the refined action, and the knowledge-routed action are summed, using a symptoms filter to avoid repeated requests. The utterances that represent the actions selected by the dialogue manager are generated by the template-based natural language generator. Multiple templates were designed for each possible action of the system, while the medical information is conveyed through the use of daily expressions related to specific symptoms and diseases extracted from a compilation of medical terms.

Xu et al. [70] proposed an end-to-end dialogue model that uses a discrete latent variable to infer dialogue intentions from the user's utterance. In the proposed system, a Bi-directional Long-Short Term Memory network is used to encode the utterance of the user. This encoding is sent to a dialogue-level Long-Short Term Memory network, along with the latent intention derived from intent network, which is modelled as a Multilayer Perceptron. This network generates a discrete latent variable that models the user's intention. The output of the intent network, the dialogue-level network and a variable indicating if there are items that satisfy user's constraints are sent to the attention decoder module, which generates the system's response. Context is maintained by a higher-level context Recurrent neural network. It processes the vector representation of each user's utterance iteratively. Thus, the hidden state of the network represents a summary of the dialogue history. This dialogue model can be trained using either unsupervised, semi-supervised, or reinforcement learning frameworks. For parameter estimation, authors proposed to apply exact maximum log-likelihood, instead of variational inference.

3.2.1.7 Markov Decision Processes-based approaches

A Markov Decision Process is a discrete-time stochastic control process that provides a mathematical framework for decision making on situations where outcomes are partially random and partially under control. Markov-Decision Processes-based approaches to dialogue management share the advantages and disadvantages of the neural network-based models: on one hand they remove the need for handcrafted rules and dialogues, but on the other hand require large datasets for training [3].

ReinForest [71] is a dialogue framework for the development of multi-domain, mixed-initiative dialogue systems. In this approach, dialogue developers need to define a knowledge ontology for all domains, while the dialogue task tree generation and execution are domain-independent. The execution of the dialogue task trees is formalized as a Semi-Markov Decision Process, thus allowing to train a traditional plan-based dialogue manager using machine learning algorithms from the field of Hierarchical Reinforcement Learning.

The core of the ReinForest framework can be divided in two modules: a knowledge ontology and a dialogue engine. The dialogue engine was designed to be domain-independent, and decides the actions of the system based on the state of the dialogue at a given point. It is divided in four components: hierarchical policy execution, belief update, tree transformation, and error handling. On the other hand, the knowledge ontology is designed by the developers, and is represented by a domain-dependent knowledge graph that holds the concepts of the domain knowledge and the

relations among them. Inside these concepts, the basic memory units are the *attributes*, which include an ID, a value and a normalised value, and a confidence score. The concepts hold an attribute map, which maps from a key to an attribute, a set of subscribed entities and domains that, when recognized in the user input, will trigger updates of the concept, and a set of dependencies among concepts. Both modules are connected by the dialogue state, represented by a pointer to the knowledge ontology and complementary dialogue information.

Two years later, Lin et al. [72] proposed a multimodal dialogue system for Conversational Image Editing. It allows users to use speech for specifying the effects that should be applied to an image. The dialogue system is modelled as a Partially Observable Markov Decision Process, and the dialogue policy is trained using a Deep Q-Network.

The architecture of the system can be divided in four modules: multimodal state tracker, dialogue manager, vision engine and image edit engine. The multimodal state tracker maintains a representation of the state of the dialogue, composed by the user state (estimation of the user's goal at a given dialogue turn) and the system state. This tracker has to store user utterances, gestures, and also the state of the image edit engine. A bidirectional Long-Short Term Memory network is used to extract the relevant information from utterances, which includes the user intent (the edit command), and the arguments that these commands require. For the gestures, slots in the state are set to 1 if gestures are present or 0 if not.

The dialogue manager uses a dialogue policy to observe the state of the system and perform the most adequate action to complete the edit operation requested by the user. These actions can be used to either request or confirm slot values with the user, or to send commands or retrieve feedback from the image edit and the vision engines. The dialogues are evaluated through two reward functions, one that takes into account the degree of success for the dialogue and its length, and another that only considers if the user goal was correctly fulfilled or not. Regarding the dialogue policy, authors compared a handcrafted rule-based and a Deep-Q-Network-based policies.

More recently, Saha et al. [73] proposed a method for incorporating the sentiment of the user during policy learning in a dialogue system for multi-intent conversations. The users proposed a two-level virtual agent modelled as a semi-Markov Decision Process, where the top level contains the intent meta-policy and the low level contains the controller policy. The rest of the system is composed of a natural language understanding unit and a natural language generation module. The intent meta-policy receives the current state of the interaction and selects the most appropriate

subtask from a list compiled based on the user requirements. The controller policy is common for all the intents of the user, thus satisfying slot constraints amongst overlapping subtasks. This policy receives also the state of the interaction and generates a sequence of primitive actions. An internal critic is in charge of giving rewards to both policies at every timestep based on the actions selected. The user sentiment is incorporated to the training phase of the agent, by combining sentiment-based rewards and the rewards generated by the internal critic. The objective is that the system is able to avoid negative sentiment during the dialogue. The sentiment is extracted using a sentiment classifier, and then incorporated into the state of the interaction and the reward models of the agent. Thus, the state space incorporates three sources of information: information about the intention, a set of confidence scores for the information slots, and the sentiment score. The controller policy will select appropriate actions to fill relevant slots pertaining to the intent in control.

3.2.2 Comparison between approaches

When comparing the works presented in the sections above, the specific issues related to human-robot interactions were taken into account. Dialogue systems for robotic applications need to manage multiple sources of information, both as inputs (e.g., the speech of the user, information coming from touch sensors...) and outputs (e.g., voice, motions, gaze...). The feature of a system concerned with this aspect of dialogue management will be defined as *multimodality* and it refers to the ability of the system to manage different sources of information. It includes both analysing multimodal perceptual information coming from the user and environment, and also being able to generate multimodal expressions that represent the actions of the system. As stated in the introduction to this chapter, speech is the primary communication channel in human-human interaction, while other modalities enhance the interaction by complementing the speech [74]. Also, while the speech is going to be the preferred interaction channel, certain situations might require the use of other modalities to overcome unexpected problems (For example, in noisy environments where the speech-based communication might fail, being able to use gestures can help the user to understand what the robot is trying to convey). Thus, being able to convey a communicative goal through different channels, or being able to fuse multiple channels to enhance the user's experience becomes a key feature in a dialogue manager for social robotics.

A second factor that has to be taken into account is the *scalability* of the system, this is, the complexity of adapting the dialogue manager to new domains. While multiple applications of social robotics can be tied to a specific domain (i.e. robots in stores, or robots for guiding people in malls, airports and other closed spaces), if a social robot is expected to be placed at the user's home and

serve as a companion, usually this is going to require the robot to be able to face multiple tasks along different domains. From a dialogue management point of view, this involves creating dialogues for all tasks and domains. Simplifying the expansion of the dialogue system to new domains and tasks becomes a coveted feature in these type of applications.

Finally, having the ability of perceiving or emitting multimodal information is an important step in providing the user with a satisfactory interaction. But being able to use multiple output sources is not enough; the information conveyed through each channel has to be combined appropriately. The system should be able to adapt the communication channels based on the particularities of the current interaction, the state of the dialogue, and other dynamic circumstances (for example, the affective state of the robot). Also, repetitive actions should be avoided, and instead, the robot's actions should present a high variability. These characteristics will be grouped under the feature called *expressiveness*. The result of the comparison between the works reviewed in this section can be observed in Table 3.1.

3.2.2.1 Multimodality

The majority of the works reviewed in this section (16 out of 23) focus on the design of a speech-based interaction system. If we group this by the approach to dialogue management selected, we can see that all the presented approaches to neural network-based dialogue management are exclusively speech-based. Regarding the works that include multimodality, we see different solutions. The approaches presented by Peltason et al. [49], Kiefer et al. [58], or Lin et al. [72] include natural lenguage understanding and generation modules for speech-based dialogues, and also can request the performance of actions using other output channels (i.e. body movements, for example) to the back-end applications. Dialogue managers for virtual agents, like the ones proposed by Wahde [54] and Morbini et al. [55] usually include animations of the virtual avatar alongside the speech or text outputs. The latter can also use visual information captured by the perception systems into the dialogue process, while in the former, user inputs are conveyed through speech or text. In the work proposed by Wessel et al. [53], the actions of the system can define multiple communication sources, while for the user input multimodal information can be stored in the dialogue act slots, alongside with the communication channel the information was received through.

Finally, the approach proposed by Alonso et al. [51] uses a fusion process to combine the semantic representation of inputs coming from multiple sources, and a fission process for output actions, where the manager selects the most appropriate multimodal action, and the multimodal fission module decouples the different modalities and sends them to the respective output modules.

Reference	Multimodality	Scalability	Expressiveness
[49]	Yes. Speech and modes provided by back-end modules	Handcrafted generic interaction patterns configured manually	Manually defined outputs
[50]	Spoken Dialogue System	Add context-related data to database and handcraft new frames and slots	Templates filled with information from the system belief
[51]	Input: aggregated multimodal info. Output: multimodal actions	New Iwaki recipes	Actions defined in the recipes, then split and sent to each channel
[53]	Only system responses	Add new ontologies and ontology-based rules for dialogue specific items	Modalities handcrafted separately. Labels in utterances replaced in runtime with proper value
[54]	Yes	Generic multi-domain dialogue building blocks configured by hand	Template-based generation
[55]	Yes. Output: speech, animation, text. Input: text, speech, visual info	Add new inference rules, event listeners	Not specified
[56]	Spoken Dialogue System	Add new set of probabilistic rules	Actions adapted through rules. Natural language generation
[57]	Spoken Dialogue System	Extend the database for the question-answering engine	Template-based approach
[58]	Yes	Rules and ontologies need to be extended	Template-based approach. Application actions predefined

Table 3.1: Comparison among the works presented in section 2. Each approach has been evaluated according to scalability, multi-modality, and flexibility.

Reference	Multimodality	Scalability	Expressiveness
[60]	Spoken Dialogue System	Add new domain knowledge. Expand the number of goals and strategies	Template-based approach
[61]	Spoken Dialogue System	Handcraft new dialogue task specification	Outputs defined in the dialogue agents. Natural language generation
[62]	Spoken Dialogue System	Modify the interaction manager and add new agents	Outputs defined in the presentation agents
[63]	Spoken Dialogue System	Define new intentions, and plans to satisfy them	Templates filled with context information
[64, 67]	Spoken Dialogue System	New dataset and new user simulator	Natural language generator
[66, 68] [70, 73, 65]	Spoken Dialogue System	New datasets for policy learning	Natural language generator
[69]	Spoken Dialogue System	New datasets for policy learning. New specific knowledge graphs	Natural language generator
[71]	Spoken Dialogue System	Expand the knowledge ontology	Natural language generator
[72]	Yes	New domain ontology. Either new rules or new dataset	Not specified

Table 3.1: Comparison among the works presented in section 2. Each approach has been evaluated according to scalability, multi-modality, and flexibility.

3.2.2.2 Scalability

This has been one of the important issues considered when designing dialogue systems, independently of the approach selected. One basic idea is present in many of the works reviewed in this section: decoupling the domain-specific aspects of the interaction from those that are task-independent, and thus can be reused when expanding the system to new domains and tasks. Different solutions have been proposed for implementing said idea. Frame-based approaches like the one presented by Souvignier et al. [50] can be extended to new domains through the creation of new frames and information slots. The dialogue manager presented by Alonso et al. [51] allows the creation of new dialogues through the addition of new Iwaki recepies. In both cases, the knowledge for conducting a slot-filling interaction is managed internally by the system, and is common for all domains. Ontology-based systems usually require the extension of the ontology with domain-specific knowledge, as well as new rules for conducting the interactions. An example of this type of dialogue managers can be seen in the works of Wessel et al. [53], Kiefer et al. [58], or Zhao [71]. Probabilistic approaches use sets of probabilistic rules to control the flow of the interaction. Thus, these sets of rules will have to be extended so they include the new domains. While the work of Lison [56] or Kiefer et al. [58] can be extended to new domains with this approach, the dialogue system proposed by Milhorat et al. [57] separates itself from the rest of information-state based approaches. In this work, the system holds a database that connects directly the user input with the appropriate response, for questions. New domain-specific elements would have to be added to the database. On the other hand, statement response strategies are multi-domain.

Following with the idea of decoupling the domain-specific and domain-independent aspects of dialogues, the RavenClaw framework propsed by Bohus et al. [61] provides a collection of general communicative abilities, while the dialogue designers focus on developing the dialogue task specification. If new domains have to be added using this framework, the task specification tree has to be expanded and redesigned. Approaches like the one presented by Nestorovič [63] require the addition of new goals and plans that can be used to accomplish the new tasks, while agent-based approaches similar to the one presented by Turunen et al. [62] can be adapted to new tasks with the addition of new agents that can manage the new domain. Finally, neural network-based approaches, as well as other dialogue systems that include machine learning solutions, will require the creation of extended datasets that represent the new domain, which can lead to scalability issues. The work proposed by Cuayáhuitl et al. [65] uses multiple neural networks to manage multi-domain interactions, having individual networks for each domain.

Finally, the dialogue systems proposed by Peltason et al. [48] and Wahde [54] have to be highlighted in this section, as they follow a similar strategy for dialogue modelling to the one considered in this dissertation: modelling dialogues as the combination of basic building blocks, that can be parametrised with domain-specific data as required. In this approach, the system can be used in new domains by building new dialogues as a combination of these basic interaction patterns and adding the domain-specific knowledge to the system's knowledge base.

3.2.2.3 Expressiveness

When analysing the expressiveness capabilities of the works reviewed in this section, two main approaches can be found. In general, approaches that fall under the category of handcrafted dialogue systems tend to use also handcrafted actions. The dialogue managers proposed by Wessel et al. [53], Souvignier et al. [50], or Peltason et al. [48] are examples of handcrafted outputs. Usually, for utterance generation, the handcrafted sentence can include labels that later will be replaced dynamically with information obtained by the system, increasing the flexibility of this expressiveness generation approach. Other works, like the ones presented by Milhorat et al. [57], Kiefer et al. [58], or Chu-Carrol [60] generate the utterances of the system based on the action selected by the dialogue manager and contextual information using a template-based approach to language generation. Finally, dialogue systems rooted in the machine learning field tend to incorporate natural language generation modules, that can either be template-based, or based on language models. The works of Saha et al. [73] and Xu et al. [69] are examples of this approach to expressiveness management. Also, in approaches that can communicate with back-end applications for executing actions, like the ones presented by Peltason et al. [48] or Kiefer et al. [58], the actions selected by the dialogue manager are sent directly to the application, which has the knowledge of how to perform said action. Other authors, like Wessel et al. [53] allow to specify by hand the individual actions for each modality when creating the dialogues.

Here, the work of Alonso et al. [51] is going to be highlighted due to the similarity between the expressiveness generation approach followed in that work and the one developed for this thesis. In Alonso's dialogue manager, multimodal actions can be handcrafted as a single item, combining all information sources. The dialogue manager selects which action has to be performed, and a separate module decides what set of individual instructions should be sent to each output module.

3.2.3 Comparison with the solution proposed in this thesis

There are a series of similarities between the works presented by Peltason et al. [48] and Wahde [54]. The dialogues are modelled using basic building blocks that represent small subdialogues. These blocks are configured using task-related information to complete certain communicative goals. In these approaches, developers handcraft the dialogues beforehand, and then the system activates them depending on the inputs received from the user, or other applications of the system. But in both works, dialogues have to be designed and added to the dialogue manager beforehand. The dialogue manager proposed in this thesis faces this issue by allowing dialogues to be built and configured in runtime by the applications of the system. On one hand, this has the advantage that only the dialogue blocks related to the tasks and inputs that the robot needs to handle when being in a given state will be active, reducing the complexity of the solution. On the other hand, this allows the developers to combine handcrafted approaches for designing dialogues for their applications with approaches that generate these dialogues automatically, as long as the basic actions in the dialogue can be represented with the building blocks provided by the dialogue manager. Regarding the management of errors during the interactions, the approach proposed by Whade require the use of specific blocks for error handling, while Peltason et al. integrate the error-handling mechanisms in all dialogue blocks. The interaction units presented in this thesis follow the latter approach, but seek to extend the range of situations that can be controlled with cases where the system is expecting to receive information from the user, but it never arrived.

The works reviewed in this section also present differences on how they manage the flow of the interaction. In DAISY, the dialogue building blocks indicate how to continue with the interaction by specifying which block should be executed next (although received inputs can move the focus to a new block). In the work of Peltason et al., the dialogue manager contains all the parametrised interaction patterns, and these are executed depending on the inputs from the user or the back-end applications. In both cases, the dialogue manager and the building blocks are the ones that define the flow of the interaction. The approach presented by Bohus et al. [61] diverges from that solution by dividing the control in two areas, where the dialogue engine provides general communicative abilities, while the structure of the dialogue is handcrafted in a higher level. But this still has the manager controlling all aspects of the interaction, and also forces developers to design their dialogues using a specific paradigm. The approach proposed in this thesis proposes a more strict division of the control over dialogues by not only separating the general communicative tasks from the structure of a particular interaction, but also taking the latter out of the dialogue manager and implementing it directly in the applications. This creates a two-level approach, where the applications are the ones that control how the dialogue has to continue, while the dialogue manager receives requests to activate

the required building blocks, controls their execution and returns the result of these sub-dialogues, while controlling that there are no conflicts between the different blocks. This allows developers to use any paradigm they want to design the structure of the dialogue, as long as the individual turns can be represented with the building blocks provided by the dialogue manager.

Regarding the expressiveness features, the proposed solution follows a similar approach to the one described in Alonso et al. [51], where the actions of the system can be a combination of multiple sources of information defined individually, or a single multimodal action that is then sent to a different module of the architecture with the expertise for dividing the action into the unimodal instructions that have to be sent to each output module. But, while in Alonso's approach the actions of the system are defined in the Iwaki recipes, in the proposed dialogue manager the actions are sent by the applications in the request used to activate a specific building block. This reinforces the idea of a two level system that externalizes the task-related aspects of the interaction. The proposed approach adds a control mechanism that ensures that the execution of the action has been completed before continuing with the dialogue, and manages situations where the execution of the action fails. Because the development of the expression capabilities of a social robot is also an important contribution of this thesis, an in depth analysis of the proposed expressiveness management architecture will be performed in Chapter 4.

3.3 Dialogue Modelling in Social Robotics: theoretical foundations

As stated by the Greek philosopher Aristotle, *"Man is a social animal"*. Humans spend their whole life in society, and tend to connect and interact with each other. Thus, the capacity for establishing communication is one of the key abilities for a person, and something that should be replicated in robots that are expected to live among humans. The question of *¿How should interactions between a social robot and a human be modelled as?* is one of the key research questions in this dissertation, and the work presented in this chapter tries to provide an answer to it from both a theoretical and technical point of view. The model designed in this thesis is rooted in the Speech Act theory [75], proposed in the context of a pragmatic analysis of language. All of these concepts are part of the area known as *philosophy of language*.

3.3.1 Philosophy of language

Philosophy of language is the branch of analytic philosophy that studies the nature of language, and its relations with the users and the world. Linguistics is the scientific discipline that focuses on the study of language, including its form, meaning, and context. While investigations in philosophy are of a conceptual nature, linguistics focuses on empirical findings. [76]. Philosophy of language has been attracting significant attention since the early 20^{th} century, although the issues started to be discussed in depth in the 1960s [77]. The philosophers in this era were driven to study language in the 20^{th} century based on the idea that it could be the path to understand the nature of reality and truth [78]. Three aspects of language were considered by most philosophers: syntax, semantics, and pragmatics. Syntax studies how words, with or without appropriate inflexions, can be arranged to show connections of meaning within the sentence [79]. Semantics focuses on the study of the meaning of language. This is an important concept in the philosophy of language, and has attracted a significant amount of attention, which resulted in the proposal of several theories of meaning. While syntax and semantics are important areas of philosophy of language, the one that is of interest for this dissertation is the area of pragmatics.

Lycan [77] defines linguistic pragmatics as *"studying linguistic expressions' uses in social contexts"*. Leech [80] proposes a similar definition: *"the study of how utterances have meaning in situations"*. From these definitions, it can be established that, while semantics is the study of what the language means, pragmatics focuses on how the language is used. This definition is also applicable when analysing pragmatics from the point of view of philosophy. Other researchers, like Bach [81], consider that the difference between semantics and pragmatics goes beyond a mere separation between meaning and use, and considers that is related to the type of information. While semantic information is encoded in the sentence uttered and the relevant data from the context, pragmatic information is generated by the act of uttering a sentence, and is relevant for determining what the speaker is trying to communicate.

During the first half of the 20^{th} century, the study of meaning attracted more attention in the field of philosophy of language than the study of language use [81]. According to Austin [75], philosophers had the assumption that the only function of an utterance was to describe the state of the world or to state facts, which could be done truly of falsely. Austin argued that there are uses of language that, although resembling fact-stating elements, are not aimed at making mere statements. In [82], four key topics are identified in the field of pragmatics. The first one is the study of deixis and indexicality [83]. Deictic systems identify the points where the linguistic structure of the sentence and the social context in which the sentence has been

used intersect. The second topic introduced is related to the concepts of reference and anaphora [84]. The former tries to find an answer to what is the relation between language and the world, this is, the relationships between linguistic expressions and the things of the world they denote. Regarding anaphora, these are expressions that refer to specific antecedents in the discourse. Thus, a theory of anaphora specifies under which conditions can anaphora uses be resolved and how these resolutions are connected to the meaning of the antecedents in the discourse. The third topic mentioned in [82] is related to inference, and its different types [85]. Inference is defined as the process of accepting a statement based on the acceptance of other statements. It involves the deduction, induction, and abduction of facts. The last topic presented in [82] has to do with the concept of *Speech Acts*. This is the one that roots the model of interaction proposed in this dissertation.

3.3.2 Speech Act theory

The Speech Act theory defended initially that there are performative utterances that do not state or describe anything, but serve to perform social acts, which are called *speech acts* [77]. This idea was later extended to point out that there is not a division between constative (sentences that describe or state facts) and performative utterances, but instead all sentences can have both aspects. Thus, a new distinction was established between two aspects of a single utterance: content and force. In particular, force refers to the intention behind the sentence. Examples of forces that a sentence can have include make a judgement, give a command, or suggest something. The original idea of speech acts was originally proposed by J. Austin in a series of lectures that he gave at Oxford and then at Harvard University. The notes of the lectures imparted at Harvard were then compiled after his death and published in the book *"How to do things with words"* [75].

According to the Speech Act theory, three aspects can be observed in utterances:

- **Locutionary act:** the sentence that is being uttered. This includes three acts: phonetic, phatic, and rhetic. The first one refers to the action of uttering sounds, the second one refers to the fact that said sounds have sense and reference, and the last one refers to the fact that the sounds uttered belong to a language, vocabulary and grammar.
- **Illocutionary act:** the action performed by uttering the sentence. It is tied to the force associated to that particular linguistic structure. It is affected by social conventions that allow a speaker to perform a recognizable action verbally. In [75], Austin presented a list of possible forces and the differences between them. He also presented the connection between these acts and the generation of effects [86]. First, both the force and the content of the sentence have to

be understood by the target. Second, the illocutionary act has to take effect. Lastly, in some occasions there might exist a need for a certain degree of cooperation from the listener.

- **Perlocutionary act:** The effect that the sentence uttered has over the world. This effect is not directly tied to the force of the sentence, and can be intentional or unintentional.

In order to distinguish these three aspects, consider the following example. If during lunch, a person sitting at a table utters *"Please, pass me the salt"*, the locutionary act would be the sentence itself, the illocutionary act would be the request performed by the speaker with the intention of getting the salt, and the perlocutionary act would be the listener giving him/her the salt.

P. Grice [87] presented a theory of speaker's meaning that changed the study of language. This theory establishes a difference between the meaning of natural events and that of intentional communicative actions performed by humans. This last category depends on the intention of the speaker when uttering the communicative action. Grice considered that linguistic meaning has to be always included in this category. A second distinction was established between speaker and sentence meaning. Thus, a communicative act requires understanding what the communicative intention of the speaker is. Searle [88], one of Austin's pupils, criticized this theory because it puts an excessive weight on the speaker's intention. Searle's theory of speech acts tries to find a balance between the views of Grice and Austin, this is, the importance of intentionality and social conventions. This theory considers that the term speech act refers exclusively to the illocutionary act of a sentence, and divides it into a illocutionary force and the propositional content, which are signalled by the illocutionary force indicator and the propositional indicator respectively. A correct understanding of both components by the listener is required to successfully completing the act (the listener has to be able to recognize the speaker's intention).

3.3.2.1 Speech Act theory in artificial intelligence and Human-Machine Interaction

Speech Act theories have left a big mark not only in the field of linguistics and philosophy of language, but also in other areas. The ones that are relevant for the work presented in this dissertation are the fields of Artificial Intelligence (AI) and Human-Machine Interaction, the latter divided into Human-Robot Interaction and Human-Computer Interaction.

Traum presented in [89] a review of relevant work in speech acts for dialogue agents, including their uses in AI. The work presented by Cohen and Perrault [90] had an important influence on the application of speech act theory in the field of AI. In this work, they modelled speech acts for

the actions of *request* and *inform* as operators in their planning system. Two preconditions are considered for these operators: that the action can be performed, and that the agent wants to do that action. In a later work, Perrault and Allen [91] used the same methodology to propose an account of indirect speech acts that can be used for requesting and informing. Their work suggests that a listener is able to recognize the action performed by the speaker, infer which goal he/she wants to achieve, and then try to help him/her to achieve this goal. Sadek [92] presented in 1992 a formal theory of intention that can be integrated in autonomous agents. It follows a logic similar to the one proposed by Cohen and Levesque for defining the semantics of speech acts. Opposite to the theory proposed by Cohen and Levesque, the one presented by Sadek is not only a meta-theory, but is instead oriented to be installed in real agents. In this approach, the communicative actions of the agent are defined based on a set of preconditions and intended effects. Vieira et al. [93] presented a work in which they added speech agent-based semantics to an agent designed using AgentSpeak, a logic-based language for programming dialogue agents. In particular, this language is used to program the practical reasoning component of the agent. The agent includes a belief base that represent the state of the world (and that can be updated), and also has access to a library of plans that indicate the actions that the agent can perform.

In the areas of HRI and HCI, one of the most common applications of speech acts occurs in the task of Natural Language Understanding known as dialogue act tagging. Dialogue act is a term that refers to the illocutionary act identified by Austin, which is equivalent to the concept of speech act as understood by Searle. It represents the intention of the speaker, and thus can be considered the basic unit of linguistic communication [94]. An example of this is the work presented by Allen et al. [95], in which they present their research on building spoken dialogue systems for HCI. The language parser installed in the dialogue system used in this work parses the input utterance to extract a sequence of speech acts. This sequence is then used for intention recognition. Another example of this is the work presented by De Carolis and Cozzolongo [96], in which the use of social robots as interfaces between users and services in a smart environment is proposed. Authors also discus how important is to recognize the attitude of the human user on top of parsing the linguistic information from the utterance. The proposed system extracts the speech acts present in the utterance, and the acoustic characteristics of said utterance, and uses a bayesian network to infer the intention of the user. The last example that will be presented here is the work of Williams et al. [97], where an experiment was designed with the objective to determine how humans used indirect speech acts in human-robot interactions, and how the ability of the robot to recognize these speech acts affected the human's perception of the robot. Their results suggest that users tend to use more indirect speech acts during task-oriented interactions, even if the robot displays an inability to understand them, that

these speech acts are more common in contexts where conventionalized social norms exist, and that the inability to understand the indirect speech acts hinders the perception that the user has of the robot, as well as how the robot performs at a task.

As shown by the examples presented, speech act theory has played a big role (and still does) in how communication can be modelled in order to design autonomous interaction systems, both embodied and virtual. This supports the initial step taken in this dissertation when designing a dialogue management architecture for a social robot, which was the identification of the basic intentions that were required in order to build task-oriented dialogues for the robot. This led to the identification of the basic communication units in the proposed system: the Communicative Acts.

3.3.3 Communicative Acts

In the context of this dissertation, a Communicative Act (from here on, CA) is the basic communication unit that will be performed during a human-robot interaction. It is an atomic element that can be configured to represent different multimodal actions, with varied intentions. Also, CAs can be combined to create more complex behaviours. Thus, whenever an application needs the robot to interact with a user, the dialogue will be modelled as a combination of CAs. The application will parametrise these CAs with task-related information required to complete a specific communicative goal. Inspiration for the CAs was taken from the concept of Dialogue Act, as each CA is supposed to represent a unique communicative intention. Theoretically, CAs can be understood as speech acts according to the definition given by Searle [88], as they represent the illocutionary acts of the dialogue. However, from an implementation point of view they are closer to the idea proposed by Cohen and Perrault [90], as they are built as a sequence of methods that seek to achieve a particular communicative goal. Based on this, *intention* became the first dimension of communication that was used to identify the basic units of interaction.

In [75], Austin presented a recollection of possible illocutionary acts and how to differentiate them, He grouped them into five main categories: *vedictive*, *expositive*, *exercitive*, *behabitive*, and *commissive*. Since then, multiple authors have proposed taxonomies for speech act classification according to their illocutionary act. The intentions selected for classifying speech acts vary from work to work, and while some works use more task-independent intentions, other rely on a bigger set of task-related speech acts. But while having this level of distinction between intentions is important when extracting information that can be used to advance a task, is not as important for modelling the external structure of the interaction. For example, Moldovan et al. separated *Wh-Questions* and

Yes/no-Questions, but in both cases, the agent has to relay some information (in this case a question), wait for an answer from the other party, and deploy any strategy necessary to counter unexpected errors that can endanger the exchange of information. Based on this conclusion, it was considered that all communicative goals can be represented using two dialogue structures: either the agent provides information without expecting an answer, or requests information from the other party. This are the possible intentions considered for the Communicative Acts, and is similar to the work of Perrault and Allen [91], where two speech acts were considered: inform and request. A combination of these two intentions, along with a proper definition of the message conveyed by the CA should be enough to fulfil more complex communicative goals. For example, if the robot has to give a command, the application can use the CA with the intention of conveying information, and parametrise it with a utterance that represents said command to transform the intention into giving an order.

The first dimension selected led to a dialogue model that allows to create interactions for accomplishing the communicative goals that a social robot should face across multiple domains. But this model would still present an important flaw: it would only account for interactions seen from the point of view of the agent, this is, it would not allow the human to take the initiative in the dialogue. This is why a second dimension had to be introduced: *initiative*. This endows the dialogue model with the ability to define which participant in the interaction is the one in charge, and will try to advance the conversation in order to fulfil his/her communicative goals. Due to the applications in which the model is going to be used, only two possibilities existed for this dimension: either the agent has the initiative, or the user has it. Because the CAs represent atomic units of interaction, there is no place for mixed initiative inside a single CA, while the model would still allow for building a mixed-initiative dialogue through a combination of CAs.

3.3.3.1 Basic CAs

The combination of the two dimensions considered (initiative and intention) led to the definition of the four core building blocks of the proposed dialogue model, as shown in Table 3.2. If the agent (in this particular case, the robot) has the initiative, then two possible CAs can be defined: *Robot Gives Information* and *Robot Requests Information*. In the former, the CA sends the message that has to be conveyed to the appropriate output interfaces, and ensures that the delivery of said message was completed successfully. For example, this would be the CA used to model an interaction where the robot greets the user. The *Robot Requests Information* CA asks a question through the corresponding output channels, ensures the correct delivery of the question, waits for a response, and checks that the information received from the user

provides a valid answer to the question. An example of this would be the robot asking the user if he/she liked a game that they just played, and waiting until the user answers *yes* or *not*. On the other hand, if the initiative is held by the user, then similar CAs can be defined: *User Gives Information* and *User Requests Information*. While the former serves as a relay that sends information provided by the user to the module of the robot that finds it relevant (e.g. the user giving a command to stop an activity the robot is performing), the latter has to also wait for that module's response and convey it to the user appropriately. For example, this CA would allow the user to ask the robot for the time. These pieces will set the foundation of all dialogues conducted by the robot.

		Initiative	
		Robot	**User**
Intention	**Giving Info**	*Robot Gives Information*	*User Gives Information*
	Requesting Info	*Robot Asks for Information*	*User Asks for Information*

Table 3.2: Description of the basic CAs that have been developed based on the two dimensions of communication considered: intention and initiative.

An argument can be made that, following the presented paradigm, any request of information is just a combination of an instance where the agent gives information (the question) and an instance where the user gives information (the answer). However, there are two reasons for why I did not follow this option. The first one is the fact that requests of information are core communicative actions that will appear multiple times in each interaction, regardless of the domain. Thus, having a CA that can represent this extremely common dialogue structure is going to simplify the design of interactions. The second reason is that, although both speakers are performing an act of providing information to the other party, the second act is constrained by the first, as it has to be an acceptable answer to the question. Having a single CA for managing both uttering the question and retrieving the answer simplifies the process of ensuring that the information obtained from the other party makes sense as an answer.

3.3.3.2 Complex CAs

With the four blocks described above, developers of applications can design interactions that achieve a variety of communicative goals without the need of having to micromanage every little detail in the dialogue. CAs also provide a set of embedded functions that are required in almost all interactions. An example of this would be the deployment of error-handling strategies. This approach gives

developers the freedom of selecting the CA combination strategy that better suits their needs. The problem with relying exclusively on four basic blocks is that, if a particular dialogue structure tied to a certain domain appears multiple times in an interaction, developers are forced to repeat the same combination of CAs, over and over. This is time consuming, and thus, a solution had to be proposed.

Because one of the key features of basic CAs (from now on BCAs) is that they can be combined with other blocks, the solution was to create new predefined dialogue structures represented as a parametrizable combination of BCAs. These new structures are known as *Complex Communicative Acts*, or CCAs. [1] While a combination of CAs can be requested in runtime, CCAs have to be handcrafted first, before they can be used in interactions. In runtime, CCAs can be requested and parametrized by any application, and combined with other CCAs or BCAs. With this approach, developers can decide which parts of the interactions that their app is going to require can be modelled as independent CCAs, and then design all dialogues either relying on these CCAs, or on the basic units. For example, if the robot asks a question during a quiz game, where the user has to provide the correct answer, the developer of the game could use a *Robot Requests Information* CA for the question and a *Robot Gives Information* CA to provide feedback, or develop a new CCA that combines both BCAs to represent all the questions in the game. This transforms interactions into hierarchical structures in which a dialogue is modelled as a combination of CCAs, which in turn represent a combination of basic interaction units (in the previous example, the game would be a sequence of CCAs for the questions in the game, while each CCA would be composed of a question CA and a feedback CA). The number of levels in this structure is not limited, meaning that new CCAs can include any combination of either BCAs or CCAs. Also, there is no limit for how many CCAs can be created, as any dialogue structure is susceptible of being described with one of these elements. The goal is to find the perfect balance between the re-usability and modularity that the reliance on basic interaction units provides, and the time saved by predefining certain dialogue sections as CCAs beforehand.

3.3.4 Division of dialogue management in two levels

The CAs presented above allow to represent short interactions between two peers. But in order to create a complex dialogue, these CAs have to be properly configured and combined. While the dialogue structures represented by the CAs are generic, and can be applied to any domain, their configuration, as well as the design of the dialogue flow, requires task-related information. This led

[1] For the sake of clarity, from now, on this manuscript will use the term CA to refer to both basic and complex Communicative Acts, while BCA or CCA will be used for each subgroup respectively.

me to propose a division of the control over the interactions between the robot and the user in the software architecture described in Chapter 2.

In the top level, applications can model an interaction as a combination of the Communicative Acts introduced in the previous section. In the bottom level, the HRI architecture loads, configures, and executes the requested CAs. Three key tasks are performed in this level: (i) Process all the information coming from the perception systems of the robot and package it into different levels of abstraction; (ii) send commands to the output interfaces to perform all the required communicative actions; and (iii) decide how to complete satisfactory the communicative goals defined by the CAs. Each of these tasks is carried out by a separate module: the Perception Manager controls all the input information processing (this module is left out of this thesis' scope), the Expression Manager is in charge of the output capabilities of the robot (this module will be presented in detail in Chapter 4), and finally the HRI Manager is the central piece in the HRI Architecture, and controls the execution of the CAs. In the proposed division of the control over communication tasks, the HRI Manager provides the blueprints for building interactions as a combination of CAs that seek to achieve different generic communicative goals, while the applications provide the task-related information necessary for properly configuring the CAs and adapt those generic goals to the specific needs of the task in hand.

3.4 Requisites identified for a social robot's dialogue manager

During the design stage of the new dialogue manager for our robotic platforms, a series of requisites that have to be met by this module have been identified:

- **Multimodality:** The proposed manager has to be able to manage multiple sources of information, both as inputs and outputs. Any combination between input and output communication modalities should be allowed. The goal is give complete freedom to the developers, so they can choose the communication channels that better suit their applications.
- **Easy configuration:** Under the proposed approach to dialogue management, the applications of the robot have to create dialogues as combinations of appropriately parametrized CAs. While the division of the control over the interactions between the applications and the proposed manager can simplify the creation of new dialogues, this advantage could disappear if the process for requesting the execution of CAs is too complicated. Thus, it is necessary to develop a standard interface between the applications and the CAs that is easy to use.

- **Conflict management:** The proposed dialogue manager has to be as robust as possible against unforeseen circumstances that might arise, either in the interaction with the user, or in the communication with the rest of the software architecture. This has been divided in two requisites. Regarding the interactions with the software architecture, the proposed manager has to include mechanisms for managing multiple interaction requests in a way that ensures that no conflicts arise. Examples of possible conflicts include the request of multiple CAs with incompatible communicative goals, or that try to use the same interfaces.

- **Error-handling:** The second requisite related to robustness involves ensuring that the interactions between the robot and the user are fluid and natural. One of the factors that will affect the naturalness of interactions is how the robot manages errors that appear during any communication (for example, problems in the reception of input information, a sudden disengagement of the user...). The implementation of the CAs has to include all the mechanisms necessary to correct some of the most common errors bound to appear in any interaction.

- **Response time:** While a proper error handling strategy can help to make interactions more natural, there is another factor that plays a role in this: the response time. The proposed dialogue manager has to be able to configure and execute CAs at a speed that guarantees that the robot is able to follow the rhythm of human communication.

Combining this requisites with the features provided by the CAs, which have been presented in Section 3.3.3, will result in a dialogue management approach able to control multimodal human-robot interactions in a way that feels natural to the users.

3.5 The HRI Manager: implementation and technical details

This section presents the HRI Manager, the dialogue manager of the proposed HRI architecture, and the library of CAs that have been currently developed. The first part of the section introduces the list of requirements that the HRI Manager has to meet. Next, the HRI Manager is presented, along with the technical aspects of its implementation, and the features that it offers. This includes the mechanisms for activating and managing CAs. Then, the library of CAs that have been developed up to this point is presented, both basic and complex.

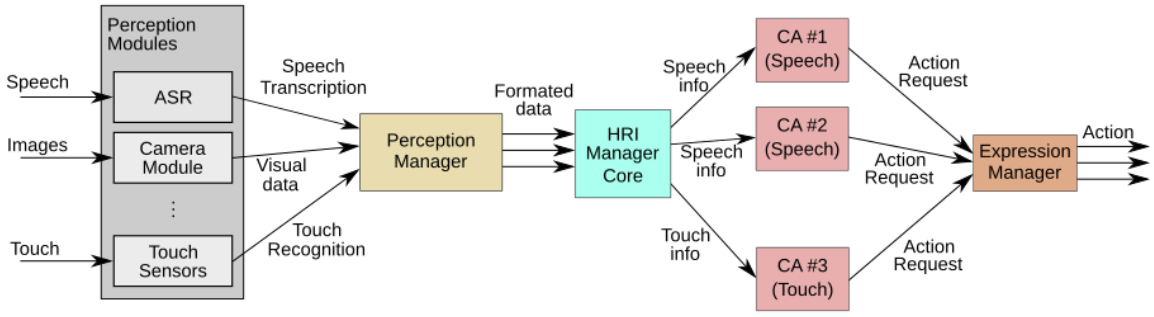


Figure 3.4: Flow of interaction-related data through the HRI System, from the information extracted from the environment by the robot's sensors to the actions performed by the robot. The parentheses in the CA blocks indicate the type of input information the CA requires.

3.5.1 The HRI Manager and its internal structure

The HRI Manager, located at the core of the HRI architecture, is the module that ensures the success of interactions between the robot and the user. It receives requests coming from the robot's applications to load and execute CAs, which serve as templates for human-robot dialogues. The information collected from the environment is sent by the perception modules to the Perception Manager, which in turn filters, formats, and packages this information according to different criteria (time window, type of data, connections between different sources of information, etc...). The packaged data is then received by the HRI Manager, and from there is relayed to the CAs that need it to complete their particular communicative goals. While the input information is filtered by the HRI Manager first so it is only sent to the CAs that were expecting it, the CAs have a direct communication with the Expression Manager. CAs can request the execution of predefined expressions or specify a set of unimodal actions that have to be performed by the robot. Flow of data through the HRI System is depicted in Figure 3.4. The operation of both the Perception Manager and Expression Manager are left out of the scope of this chapter.

In the proposed software architecture, all communications between modules are handled using ROS [98], a middleware for robotics that provides a communication layer on top of the operating system in order to interconnect an heterogeneous collection of software modules.

From a structural point of view, the HRI Manager can be decomposed into two main elements: the HRI Manager core and the CA Library. These elements are shown in Figure 3.5. The HRI Manager core serves as a hub that connects the perception architecture and the applications of the robot with the CAs. It is in charge of processing all requests for CA activation, managing conflicts that

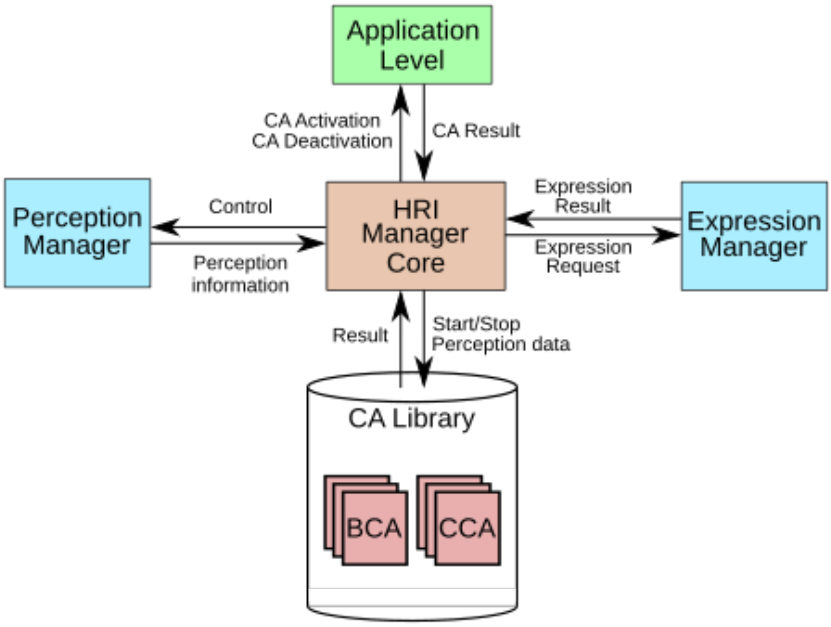


Figure 3.5: Schematic view of the HRI Manager, depicting the relation between the HRI Manager core, the CAs, and the other modules of the software architecture.

might arise among CAs, and filter all the information coming from the Perception Manager so only the relevant one is sent to the CAs. This core is designed as a single process that combines a control loop where CAs are loaded, parametrized, and executed, with a series of callbacks for managing the reception and formatting of information coming from different parts of the software architecture. In particular, this module takes care of the following tasks:

1. Manage CA requests.
2. Configure the perception modules.
3. Execute the requested CAs.
4. Manage input information.

An activity diagram representing the relationship between these steps is shown in Figure 3.6. All of these tasks will be presented in more depth in the following sections. Figure 3.7 shows the class diagram for the HRI Manager and the CAs. The HRI Manager core has been modelled as an individual class, named HRIManager. This class can instantiate objects from the CAThreadBase to create individual execution threads for each CA. Each of this threads in turn instantiates an object from either the ContinuousCA or the ImmediateCA class. These classes manage the loading

and execution of CAs. In the software architecture shown in Figure 3.7, all the CAs that have been developed inherit from a common template: the CommunicativeAct class. In the proposed architecture, the CAs are represented as state machine-like structures that include different states (for example, a state can be used for sending info to the user, while another can be used for waiting for a gesture to end). All states developed inherit from a common class, called BaseState. All the elements and processes mentioned here will be presented in depth in the following sections.

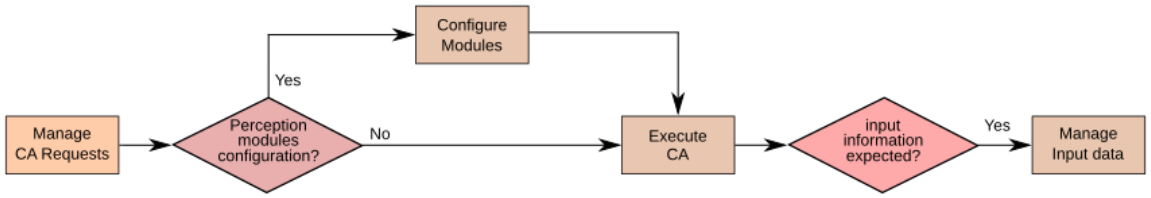


Figure 3.6: Tasks that the HRI Manager core has to complete.

3.5.2 Managing CA requests

All the interactions that the robot has to conduct are designed as a combination of adequately parametrized CAs. Applications can send requests for the activation of specific CAs to the HRI Manager, along with any task-related information required to configure it (e.g. the grammar that will be used to recognize the user's utterances). The HRI Manager core receives these requests, and decides how the CA will be executed. While some CAs are incompatible, meaning that they cannot be executed at the same time (for example, the robot cannot and should not ask a question and provide unrelated information at the same time), others can be performed concurrently. The latter is useful for example for managing different conversation topics at the same time. Combining the execution of different CAs is not a trivial process, as the clash of two incompatible CAs could result at best on an interaction that does not feel natural (for example, if the robot starts switching between topics and communicative actions incoherently), and at worst in a conflict that puts the whole operation of the HRI Manager at risk (for example, several CAs trying to access the same system resources at the same time, causing some of them to crash). Thus, the most important decision that the HRI Manager core has to make when processing CA activation requests is to decide which CAs can be executed immediately, which should be ignored, and which should just be kept in store to be executed when possible. The solution to this problem was to develop a priority system that could be used to compare the importance of performing each CA.

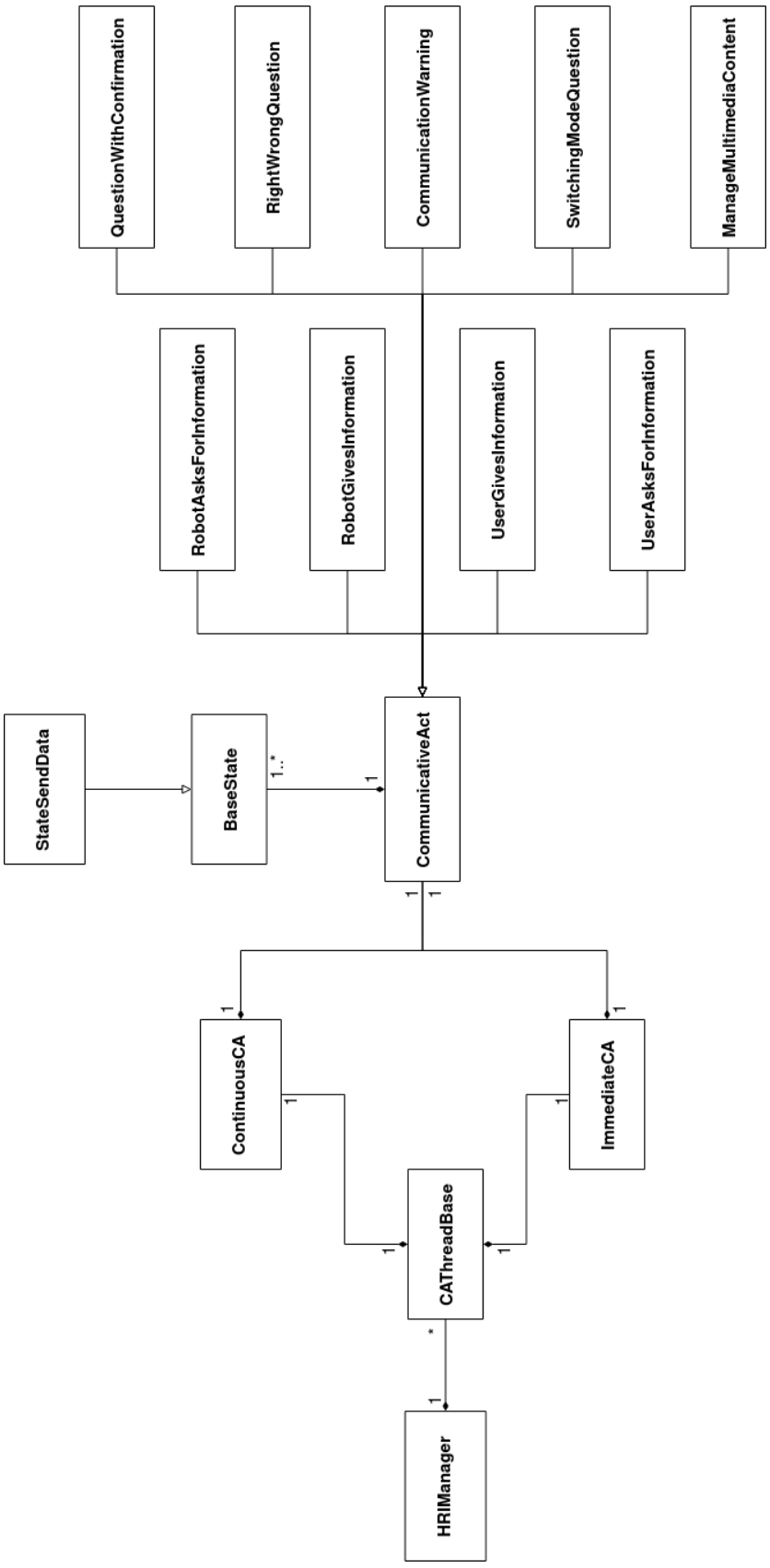


Figure 3.7: Class diagram representing the relationships between the HRI Manager core and the library of CAs. The list of attributes and methods for each class can be found in Appendix B.

The proposed system considers three different levels of priority: low, medium, and high. By default, applications will use the medium level of priority for their CAs. It can be considered that there will never be a situation where multiple applications try to push different communicative goals through the request of medium priority CAs. This is because the DMS controls the activation and deactivation of applications. If a situation that requires an immediate response from the robot arises, the application in charge of managing said situation can send high priority CAs in order to ensure the immediacy of the robot's response. Finally, non-essential abilities of the robot are restricted to moments where no task is being conducted, and thus they will request CAs with low priority. The operation of the priority system can be better understood through the following example. A user wants to engage in a game with the robot. In this situation, all the interactions conducted in the context of the game will have medium priority, as the application in charge of games is the one driving the robot behaviour at the time. While the robot and user play the game, a scheduled event requires that the robot notifies the user that it is time for him/her to take his/her medicines. Because this is a critical notification, it will be pushed with high priority to ensure that the user receives the message as soon as possible. Finally, while the robot is able to answer basic questions, like the time of day, this ability is considered to be non-essential, and thus the CAs coming from the application that provides answers to these questions will have low priority, so they do not interrupt the execution of the game.

Besides priority, there is a second factor that plays a part on deciding which CAs can be executed and which should be ignored: which speaker is holding the initiative in the interaction. This is due to the fact that, while the system always knows which interaction goals the robot wants to achieve at any given time, what the user might want to do cannot be accurately predicted. This led to the design decision of assuming that two CAs that manage interactions where the robot has the initiative would never be active at the same time, as it makes no sense to have the robot trying to achieve two completely unrelated communicative goals at the same time. This does not mean that the robot cannot try to obtain multiple pieces of information at once, but instead that there can only be one end goal. For example, while a communicative goal can be obtain personal information about the user, which could be obtained through a single CA that asks for his/her name, surname, and date of birth, there will never be a situation where the robot tries to achieve at the same time the goals of ascertaining these facts about the user and giving him a weather forecast, for example.

On the other hand the system needs to be ready for the user to try to achieve one goal amongst many at any time, which is done by keeping multiple CAs where the user has the initiative active at the same time.

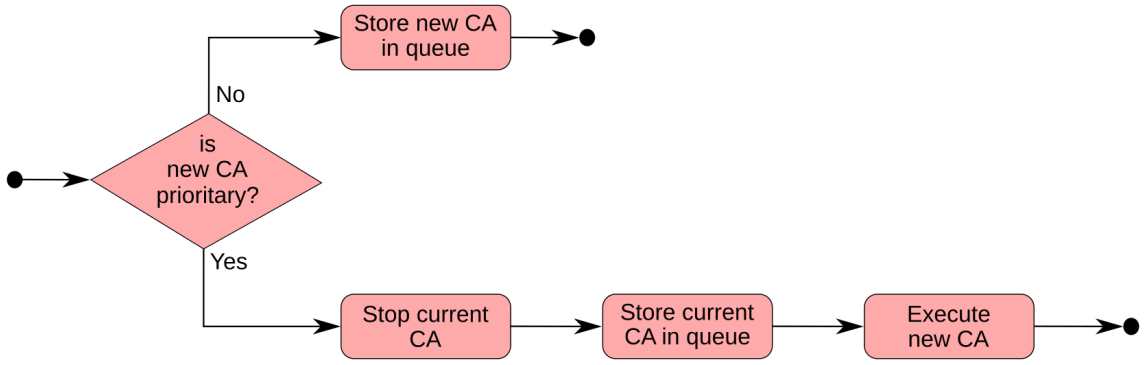


Figure 3.8: Activity diagram showing the priority management strategy for CAs where the robot has the initiative.

The HRI Manager core maintains individual queues for each priority level. If an application requests the activation of a CA to manage a situation where the robot has the initiative, then the subscriber in charge stores this request in the appropriate queue depending on the priority level defined in the activation request. In this architecture, the priority queues follow a first-in, first-out approach, where the coincidence of CAs with the same priority is solved by giving more importance to the CA that was requested first. If the new CA requested has a higher priority that the one that is currently being performed, then the HRI Manager core stops what the robot is doing, handles the urgent situation immediately, and then goes back to the CA that was being executed. This is achieved by storing the CA that has been stopped at the beginning of the corresponding priority queue, instead of at the end. The process for managing priorities for CAs where the robot has the initiative is shown in Figure 3.8

Regarding the CAs managing situations where the user has the initiative, priority queues are no longer required, as these CAs can be executed immediately. The only issue is ensuring that the activation of a new CA does not cause a conflict with regards to the usage of the robots interfaces. For example, if two CAs need to use the same touch sensor to retrieve information, then it is impossible to ascertain to which CA belongs the information coming from the perception modules. This limitation is not tied to the usage of specific communication channels, but instead to the ability of the HRI Manager to discern to which CA should be sent the information coming from the user. For example, while it has been established that two CAs cannot share the same touch sensor, they can both wait for speech-based information, as the distinction between CAs can be established based on the semantic information extracted from the speech. Based on this, when a request for activating a CA that handles user-initiated interactions is received, the HRI Manager core checks if it is safe to

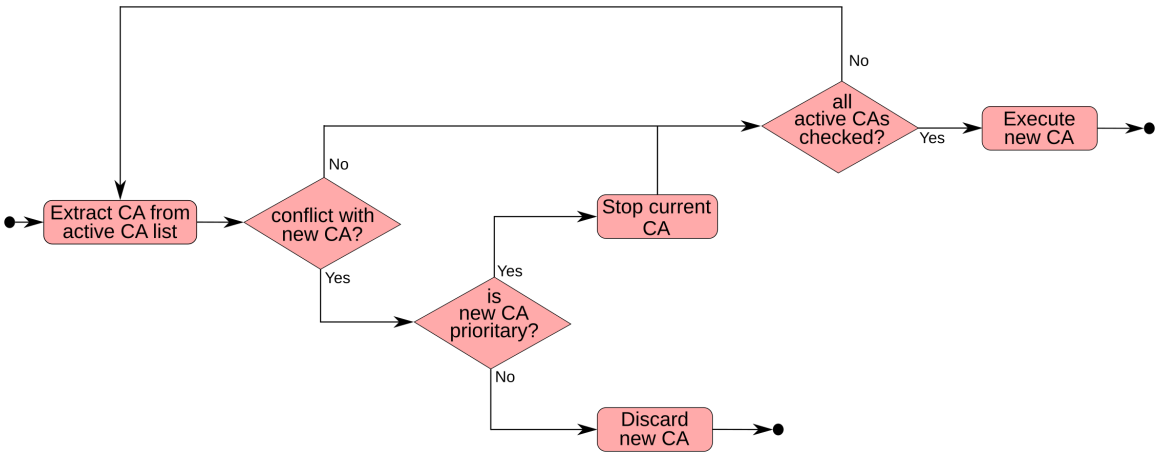


Figure 3.9: Activity diagram showing the priority management strategy for CAs where the user has the initiative.

execute the CA immediately. In the event of a conflict between the new CA and one of the active ones, then the one with the higher priority is left active, while the other is discarded. The process for managing priorities for CAs where the user has the initiative is shown in Figure 3.9

The last situation that has to be considered is the possibility that a CA where the robot has the initiative and one where the user has it require the same interfaces. An example would be the robot asking the user to touch its shoulder in the context of a game, and a CA that allows the user to stop what the robot is doing by touching that same shoulder. A practical solution was found to solve this problem. CAs that allow the user to take initiative are usually active for extended periods of time, and might end not being used at all. On the other hand, CAs where the robot has the initiative are only active for short periods of time, and are requested when the robot needs to complete a specific task immediately. The solution implemented was to consider CAs that handle robot-initiated interactions as having a higher priority, as it is preferable that the robot ignores temporarily some of the user's actions over skipping one or more of the robot's actions, which could end up affecting the applications negatively.

3.5.3 Configuration and execution of CAs

In the HRI Manager core's control loop, the priority queues are checked once every 0.1 seconds. This frequency was selected in order to balance the time constraints involved in human interaction and computational load. The status of the queues is checked in order of priority, from high to low. If

a request is found in one of the queues, then the HRI Manager analyses the information contained in that request and prepares the execution of the CA. First of all, the appropriate template has to be loaded from the CA library (see Section 3.5.6 for a detailed description of all the available templates). In the proposed module, CA templates are loaded as individual configurable software modules. Once the appropriate template has been loaded, the next step is to configure it properly, so it reflects the communicative goal that the application wants to achieve. For this, two types of configuration parameters can be found: generic and CA-specific. Generic parameters are common to all CAs, and used by the HRI Manager core to build all the structures required to execute and control the CA, while CA-specific parameters include all the task-related information.

The generic parameters include:

- **Name:** String used to uniquely identify the CA. This is necessary due to the fact that there might be more of one instance of a given CA template in use at the same time, and the HRI Manager core needs to control them independently. For example, the name of the CA will be used when defining the communication channels between the CA and the HRI Manager core.
- **Type:** The name of the CA template that has to be loaded
- **Priority:** The priority level that will be assigned to the CA in order to solve any potential conflict that might arise with other CAs, as presented in the previous section. Three possible values: high, medium, and low.
- **Emitter:** The name of the application that is requesting the activation of the CA. This will be used to return the result of the interaction directly to that application.
- **Duration:** This parameter allows to define the condition for finishing the execution of the CA. It will be defined more in depth in the following paragraphs.

There are two possible configurations for the duration parameter: *ending* or *continuous*. If a CA is configured as *ending*, then it is executed once and then discarded, after sending the result of the interaction (if it was successful, a failure, or if it was cancelled), along with any information retrieved by the CA that might be of interest to the application. *Continuous CAs* are executed in a loop, and will remain active until an explicit deactivation request is sent by the application that requested its activation, or any other software module with knowledge of the CA's name. Every time the interaction described by the *continuous* CA is completed, the result is sent to the application, just like in the case of *ending CAs*, but instead of discarding it, the CA is executed again from the beginning.

There is no restrictions about which CAs can be configured as *ending* or *continuous*. Nevertheless, there is a criteria that is followed in the majority of situations: CAs where the robot take the initiative will be configured as *ending*, while CAs where the user is the one taking the initiative will be configured as *continuous*. This criteria has its roots in the fact that applications know the time instant in which a robot-initiated interaction should start, and there are not many reasons for constantly repeating a particular interaction, while user-initiated interactions are unexpected, and can be required more than once. An example of the latter would be a CA that allows the user to issue commands to the robot. The CA would stay active for as long as those commands have to be controlled, and it might be necessary to manage more than one command. Meanwhile, if the robot needs to remind the user about a scheduled appointment, then the CA can be configured as *ending*, as this interaction only has to happen once, at a particular moment in time.

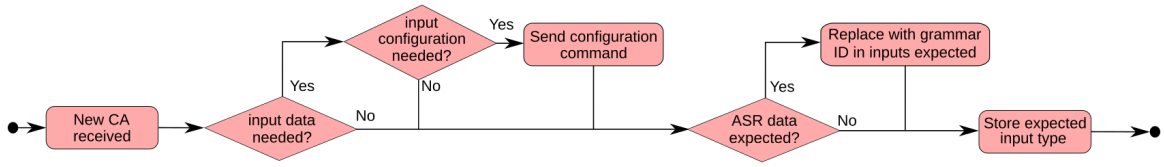
While the parameters described above have to be present in every CA, there is also information that is connected to the communicative goal that a particular CA has to achieve, and thus might not be essential for all CAs. In general, these CA-specific parameters are connected to one of two tasks: sending information to the user, or configuring the reception of the user's response. Parameters related to the first task include not only the message that has to be conveyed to the user through one of the available communication channels, but also any other parameter needed to configure those channels (for example, the distribution that has to be used to arrange multimedia content that will be displayed in a touch screen). Parameters related to the second task involve both the configuration of the input modules of the robot (for example, displaying a menu on a touch screen so the user can select an option, or loading a grammar in the grammar-based ASR) and also the configuration of how the CA will retrieve information coming from the user. This includes defining which input channels can be used, how long will the CA wait for an answer, or what is the correct answer (if the CA is used to ask a question that has a correct answer). While most CA-specific parameters are handled internally by the CA, the configuration of the input interfaces of the robot is performed by the HRI Manager core during the parametrization of the CA template.

Once the CA template has been loaded and properly configured, the HRI Manager core starts its execution. Each CA is run on a separate thread in order to obtain a concurrent execution. Whenever the interaction described by a CA is completed, the result is relayed first to the HRI Manager core and then sent from there to the application that requested the interaction in the first place. This was designed as a two-step process in order to centralize the communication between the high level of the software architecture and the HRI Manager. Also, at any given time, applications can send requests to deactivate any CA currently in execution or stored in one of the priority queues. The deactivation

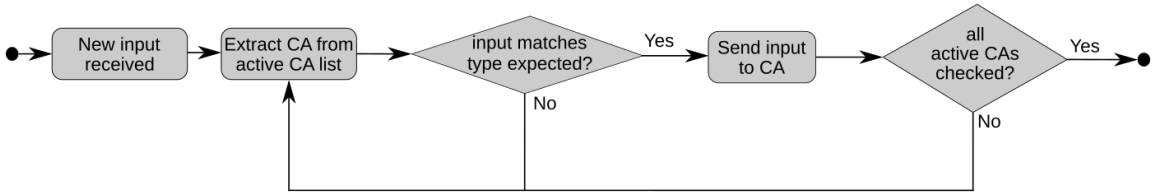
request contains the name of the CA that has to be cancelled. If it is active, the HRI Manager core sends a cancellation command that leads to the preemption of the CA and notifies the application. If it is not active, but the CA is stored in one of the priority queues, then it is simply discarded.

3.5.4 Processing input information

As stated in Section 2.3.3, in the software architecture of the robotic platforms used in this thesis, the information captured by the perception modules of the robot is packaged in different levels of abstraction. In particular, although three levels have been proposed, only two of them have been currently implemented. The HRI Manager core uses individual callbacks for processing the information coming from each level of the Perception Manager. In both cases, the inner workings of the callback is identical. When a CA is ready to be executed, the HRI Manager core checks if any information coming from the environment will be required. If so, the name of the CA and the type of information expected is stored in an individual dictionary for later use. Information type is not restricted to the communication channel used to retrieve the information, but instead the feature used to connect this information to a particular CA. For example, while all the detections captured by the robot's touch sensors are sent to all the CAs expecting touch information, speech recognitions are only sent to specific CAs based on semantic information provided by the ASR.



(a) Activity diagram representing steps taken during the configuration of a new CA for managing input data.



(b) Activity diagram representing the process followed when new input data is received.

Figure 3.10: Activity diagram representing the management of input data by the HRI Manager core.

Whenever the Perception Manager sends new input data from any of the two abstraction levels, the HRI Manager core has to check which CA/s can find that particular information useful.

Because CAs can be expecting information to come from multiple sources, or even a combination of perception data, the HRI Manager core has to ensure that all required information types are present in the message received from the Perception Manager before relaying it to a CA. For example, if a CA is waiting for a combination of touch and speech-based perception data, then the HRI Manager core will only send the perception messages that contain both types, while ignoring those that contain only one of them or neither. First, the perception message is processed so it fits the standardised structure expected by the CAs, and then sent to all CAs expecting it. Figure 3.10 shows the steps followed by the HRI Manager core for configuring CAs expecting input information and for managing new inputs received from the Perception Manager.

3.5.5 Creation of new CCAs

In the proposed approach, CCAs can be developed through two different methods. The first one involves manually coding them, following the same process used to develop the BCAs. This task can be time consuming, and requires a certain level of knowledge about the internal mechanisms of the system and the libraries used to develop the CAs. In order to simplify the creation of new CCAs, a second method was designed: define the structure of a CCA using markup language. This type of CCAs have been called *Frequent CAs*. They are identified by an unique name, which will be used to request their activation, and contain a sequence of sub-CAs, as well as the parameters needed to configure them. These parameters include the name (which has to be a number, starting in 0 and increasing one by one), the type and the parameters needed to configure the inputs and outputs to the sub-CA, as the rest of generic parameters (emitter, priority, and duration) are taken from the activation request. Also, each sub-CA has an extra parameter called *transitions*, which is used to define which of the sub-CAs should be executed after the current one has been completed.

Whenever the HRI Manager extracts a CA from one of the priority queues, it checks if the type of the CA is included in the list of available *Frequent CAs*. If it is, the list of interfaces used by all the sub-CAs is added to the activation request, and then the HRI Manager analyses the conflicts that could arise between the *Frequent CA* and the rest of active CAs, following the procedure described in Section 3.5.2. Then, the HRI Manager loads the first sub-CA from the *Frequent CA* template (the first CA is the one named 0) and executes it. When the execution of the sub-CA is completed, the result is used to select the proper transition and start the execution of the next sub-CA. Transitions can be connected to the result of the sub-CA (if it has been successful or not) or to information retrieved during the interaction (for example, have a sub-CA for asking a yes/no question, and then move to one of two sub-CAs depending on the answer). If the transition leads to an outcome

(*succeeded* or *failed*) instead to a new sub-CA, then the execution of the *Frequent CA* is completed and the result is sent to the applications, like for any other CAs.

3.5.6 Library of Communicative Acts

In the proposed architecture, CAs are modelled as structures similar to finite state machines, but with extended capabilities (for example, the possibility of having concurrent states). The technical implementation of the CAs has been done using SMACH, a python library for a fast prototyping of hierarchical state machines [2]. The library of CAs includes both the templates developed for each individual CA and a collection of the states that will be combined to create new CAs. From a programming perspective, all CAs are inherited classes of a common template that provides the implementation of the mechanisms for cancelling the execution of the CA, as well as the creation of the communication channel required to use this feature. This common template also provides tools for creating logs of the interactions. States also inherit from a common state template that implements a single method for sending commands to the output interfaces of the robot. This is necessary because there are multiple states that need to be able to send actions to the robot's expressiveness. On the other hand, inputs will only be received by a single state.

This section introduces all the CAs that have been currently develop, starting with the four BCAs that were presented in Section 3.3 (*Robot Asks for Information*, *Robot Gives Information*, *User Asks for Information*, and *User Gives Information*), and then continuing with all the complex CAs (or CCAs). While the set of BCAs is closed and was defined based on the combination of the *initiative* and *intention* dimensions, the number of CCAs is not fixed, as they have been identified empirically during the interactions with the robot, and thus is possible that new CCAs are found in the future. Also, although from a theoretical point of view it is stated that CCAs are combinations of the four BCAs (and possibly other CCAs), in reality they are a combination of BCAs with other states used for controlling the flow of the interaction (for example, a state that receives the result of one BCA, and based on that decides which is the next BCA that should be executed).

The robotic platforms used in this thesis have been designed to, among other tasks, propose cognitive stimulation exercises to the users (the robot is targeted to older adults that suffer from mild cognitive impairment). Considering the application of these robots and the requirements that these exercises have, the following CCAs have been identified: *Right-Wrong Question CCA*, *Question*

[2] https://wiki.ros.org/smach

with Confirmation CCA, *Switching Mode Question CCA*, *Manage Multimedia Content CCA*, and *Communication Warning CCA*.

3.5.6.1 Robot Gives Information

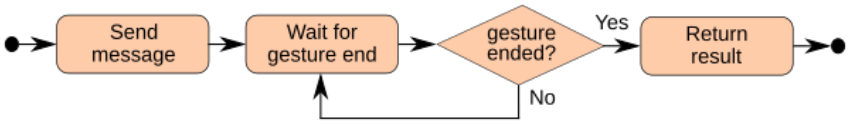


Figure 3.11: Activity diagram for the Robot Gives Information CA.

This CA, shown in Figure 3.11, is used to model interactions where the robot initiates a dialogue with the goal of conveying information to the user the user. The robot can deliver this information through any communication channel available, or even through multiple channels at once. The CA-specific parameters include the message that has to be emitted, as well as all necessary information for configuring the output modules of the robot. After the message has been sent to the output interfaces of the robot, the CA waits for the action to be completed, and then returns the result of the interaction, which depends on the success or failure of the expression sent to the Expression Manager. For example, if the robot has to inform the user that is time to take his/her medicines, the utterance is sent to the Expression Manager, which in turn relays it to the TTS module (the inner workings of the Expression Manager will be introduced in the next chapter of this manuscript). Then the CA waits until the Expression Manager notifies that the etts has finished uttering the sentence, and returns the result to the HRI Manager core.

3.5.6.2 Robot Asks for Information

The robot seeks to retrieve information from the user. This includes asking questions or giving commands that involve a response from the other peer. Similar to the *Robot Gives Information* CA, the communicative action of the robot can be displayed through one or more interaction channels. In this case, the response of the user can also come through different input channels (the interfaces that the user can use are defined in the CA-specific parameters). The user can be given the choice of which interaction channel to use, or he/she can be requested to use a specific one, or even a combination of them.

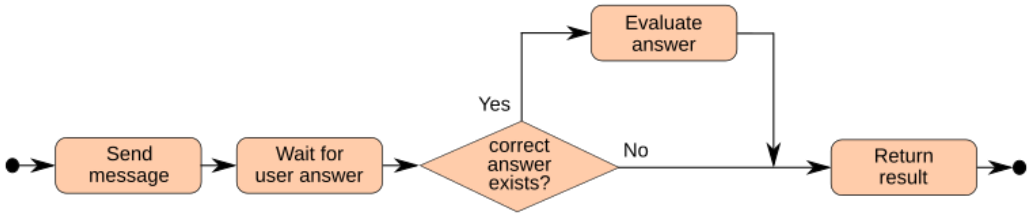


Figure 3.12: Activity diagram for the Robot Asks for Information CA.

Figure 3.12 shows the activity diagram for this CA. The structure of this CA template can be divided in either three or four stages, depending on the necessity of providing a specific answer. The three stages that will be always present are: (i) sending the robot's action to the Expression Manager and waiting for its completion; (ii) waiting for the user's response, which involves ensuring that the answer received meets the CA's expectations (comes through the appropriate channels); and (iii) return the result of the interaction to the HRI Manager core. If there is a correct answer, then an extra stage is required to check that the answer given by the user is in fact the correct one. Finally, while the HRI Manager core configures the input modules as required by the CA, if the user's answer has to be collected through a menu, it will be sent to the Expression Manager alongside the expression that the robot has to perform, and displayed once the execution of this expression has been completed.

3.5.6.3 User Gives Information

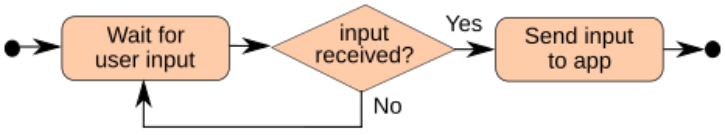


Figure 3.13: Activity diagram for the User Gives Information CA.

This CA template allows the user to communicate any information to the robot. Because the CA has to be requested by one of the applications, this ensures that the robot will only pay attention to those commands that can be processed. This CA assumes that the user is not requesting any information back from the robot, although the command of the user itself can have a visible effect on the interaction. For example, if the user asks the robot to stop a game they are playing, the cancellation of that game could be considered a response to the request of the user, but this response does not involve the transmission of information.

The activity diagram for this CA can be seen in Figure 3.13. Once the CA has been activated, it waits indefinitely for a command of the user. If one arrives through the appropriate channels, then it is relayed to the application that requested the activation of the CA, and the execution of the CA ends. If the application requires that the CA remains active in order to process more than one command, then it should configure it as a *continuous* CA.

3.5.6.4 User Asks for Information

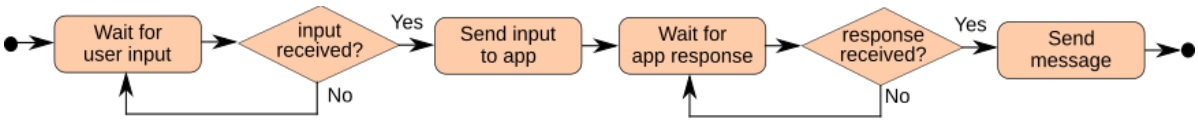


Figure 3.14: Activity diagram for the User Asks For Information CA.

This CA, shown in Figure 3.14, manages all interactions initiated by the user where the objective is to obtain information from the robot. From a conceptual point of view, the effect of this CA would be similar to the combination of the *User Gives Information* and *Robot Gives Information* CAs. The *User Asks for Information* CA waits indefinitely for the user to issue a command, then relays it to the proper application, and waits for a response. The application gathers the information requested by the user, and sends it back to the CA, which in turn sends it to the Expression Manager to be conveyed. Once the information has been properly expressed, the execution of the CA is completed. An example of a possible use for this CA would be an application that provides answers to frequent questions that users might have, like the current time, or date.

3.5.6.5 Right-Wrong Question CCA

As stated before, one of the key applications of the robotic platform used in this thesis consists on a series of cognitive stimulation exercises in which the robot proposes a series of questions to the user oriented to stimulating different areas of cognition (memory, perception, etc...). These exercises showed that there is a need for a series of features that are not covered by the basic *Robot Asks for Information CA*. Thus, the *Right-Wrong Question* CCA, shown in Figure 3.15, was developed to include these capabilities.

Among the new features, this CCA is able to give feedback to the user after an answer was provided for a question that has a specific correct answer. The CCA would congratulate the user if the answer is correct, or would encourage him/her if it was wrong. A second feature included in this

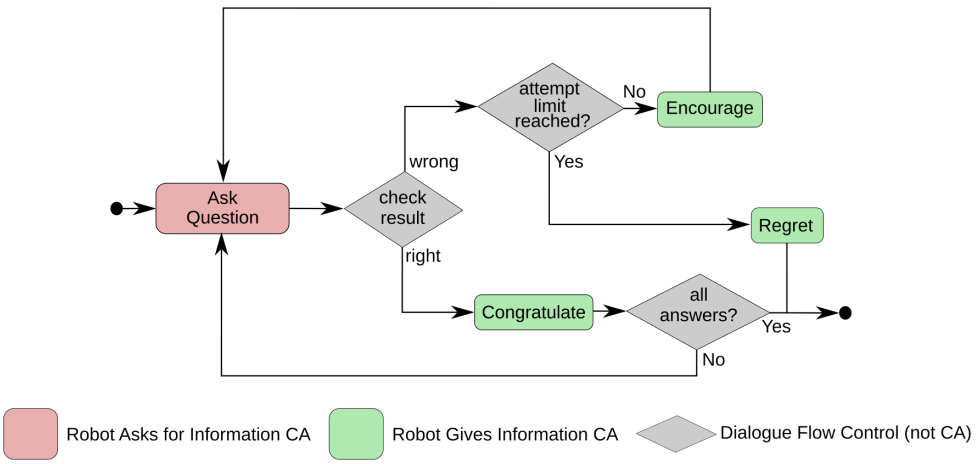


Figure 3.15: Activity diagram for the Right-Wrong question Complex Communicative Act (CCA). In this example, the CCA expects a sequence of answers in a given order.

CCA is the ability to give the user multiple attempts to provide a correct answer. This would affect also to the feedback provided by the robot, as the CA would encourage the user to answer again if a wrong answer was given but the attempt limit has not been reached, or would try to cheer him/her up if there are no more attempts. A third feature allows this CCA to ask for a single response (e.g. *What is your name?*), or for a series of answers (e.g. *Which are your three favourite TV shows?*). In the case of multiple answers, the user can be given the freedom of choosing the order in which they are given (an example would be the previous question about the user's preferences on TV entertainment), or might instead be required to provide the answers on a predefined order (for example, asking the user to repeat a list of words that the robot said to him/her as a part of a memory game).

The basic configuration of this CCA (a question that looks for a single answer) parametrizes a *Robot Asks for Information* CA with the question that has to be asked to the user, and a series of *Robot Gives Information* CAs to manage all the possible variations of the feedback that the robot can provide. The CCA would select one of these feedback CAs depending on the result of the *Robot Asks for Information* CA (if the user's answer was correct or wrong). If more than one answer is required, then the CCA has to configure as many *Robot Asks for Information* CAs as answers are required (thus, a question that seeks multiple answers would be modelled as a sequence of questions that seek for an unique answer each). They will be connected in a sequence, using *Robot Gives Information* CAs to connect them (the robot would utter something similar to *"Correct, go on"*, or *"Perfect, let's continue"*). If the answers have to be given in a particular order, each *Robot Asks for Information* CA will have only one correct answer (the first CA will look for the first answer in the sequence, and

so on). On the other hand, if the answers can be given in any order, then the first *Robot Asks for Information* CA will accept any of the expected answers, while for the next ones the answers already obtained would be removed from the list of expected answers. In any case, once all the answers are obtained, the CCA gives feedback to the user similar to the one given in the case of a single answer.

3.5.6.6 Question with Confirmation CCA

There can be many unexpected issues that arise during an interaction and that might hinder the ability of the robot to communicate with the user. In speech-based interactions, an error during the recognition of the user's speech might result on an interruption of the communication. Having a strategy for correcting the recognition of the user's speech can help to improve the quality of the interactions. This is the goal of the *Question with Confirmation* CCA, shown in Figure 3.16.

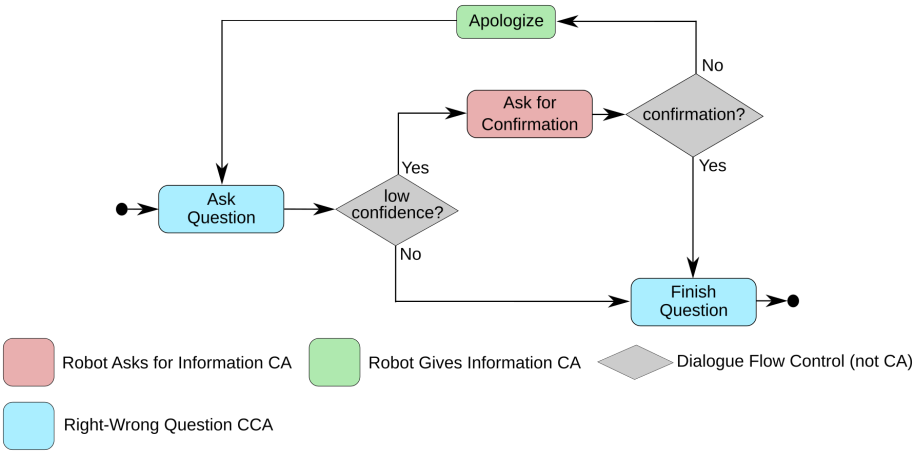


Figure 3.16: Activity diagram for the question with confirmation CCA. This CCA asks the user for an explicit confirmation if the confidence of the answer recognition is below a given threshold.

The CCA configures a *Robot Asks for Information* CA to ask a question to the user. If the ASR module of the robot cannot produce a recognition of the user's speech with a level of confidence that is high enough, the CCA would use a different *Robot Asks for Information* CA to ask for an explicit confirmation of the result of the recognition. If the user confirms the recognition, then the question continues its execution normally. On the contrary, if the user rejects the result provided by the ASR, then a *Robot Gives Information* CA would be used to issue an apology, and then the question would be repeated.

3.5.6.7 Switching Mode Question CCA

While asking for an explicit confirmation might be a possible solution for an unreliable recognition of the user's speech, it is not applicable to other communication issues that might arise during an interaction (for example, an absence of a recognition). Thus, having a backup plan in case the primary interaction channel fails can be another effective way of correcting communication problems.

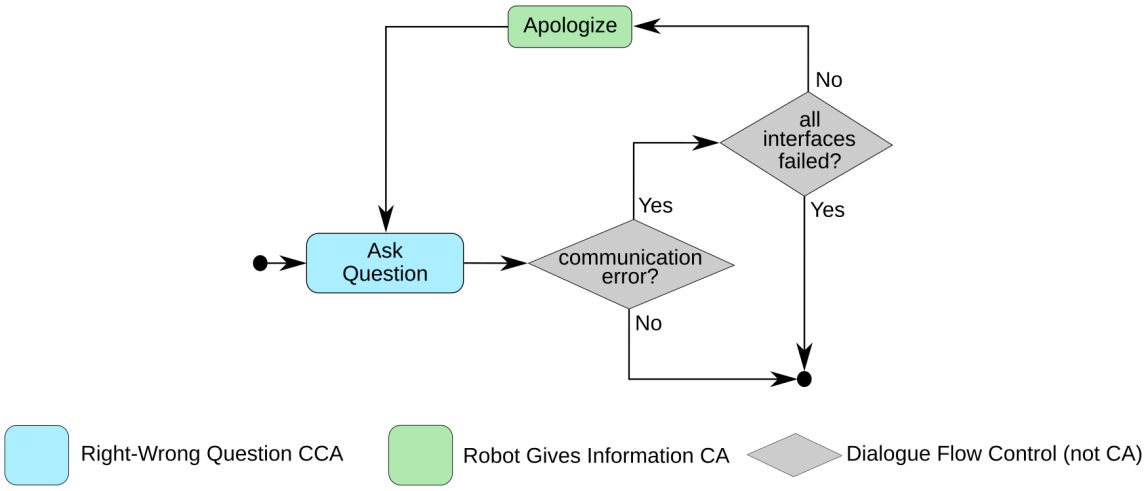


Figure 3.17: Activity diagram for the switching mode question CCA. This CCA can be configured with multiple input modes, and then it switches from mode to mode in case of communication problems.

The *Switching Mode Question CCA*, shown in Figure 3.17, allows to configure multiple versions of the same question, each of them using a different input channel. The application can specify the order in which the channels should be tried. If communication using the primary channel proves to be impossible, then the CCA switches to the next modality. The process is continued until one of the questions can be completed successfully, or until all modalities have been tried without success. For this CCA, instead of using the basic *Robot Asks for Information* CA to obtain information from the user, the *Right-Wrong Question* CCA is used. This demonstrates the modularity of the proposed dialogue model, where BCAs can be combined to obtain Complex CAs, which in turn can be combined into more and more complex interactions.

3.5.6.8 Manage Multimedia Content CCA

One of the application included in the robotic platforms used in this thesis can use the robot's touch screen to display multimedia content, like videos, music, or text. The application needs to provide the user a way to control the content being displayed (pause, stop, or resume it). While this could be achieved with a proper configuration of BCAs, it was deemed to be a structure common enough to be modelled as a CCA.

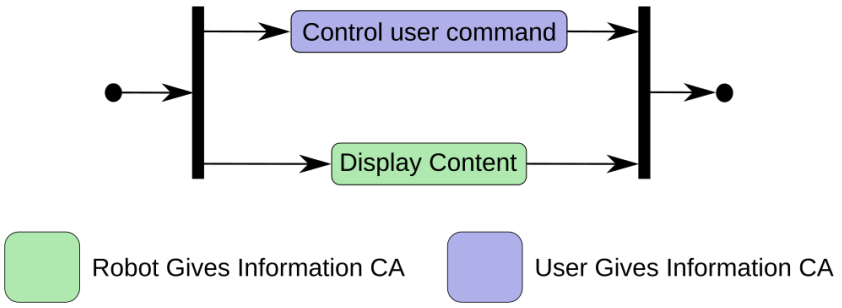


Figure 3.18: Activity diagram for the manage multimedia content CCA. This CCA sends the content to the touch screen and allows the user to control the execution of the content with voice commands.

The *Manage Multimedia Content* CCA, shown in Figure 3.18, executes concurrently two BCAs: a *Robot Gives Information* CA and a *User Gives Information*. While the former is used to send the multimedia content that has to be displayed to the tablet, the latter relays all the commands coming from the user to the multimedia application. It is important to mention that the functionalities related to pausing, stopping, and resuming the multimedia content are not included in the CCA, but instead are managed directly by the application.

3.5.6.9 Communication Warning CCA

The last CCA developed was created with the goal of protecting the system from failures in the robot's hardware, specifically the elements that provide the communication capabilities of the robot, both perceptual and expressive. It is the only CA that is not used by the applications, but instead managed internally by the HRI Manager core. Whenever the connection between the HRI Manager and one of the robot's input or output interfaces is broken, the *Communication Warning* CCA is loaded and parametrized with the information related to the interface that failed.

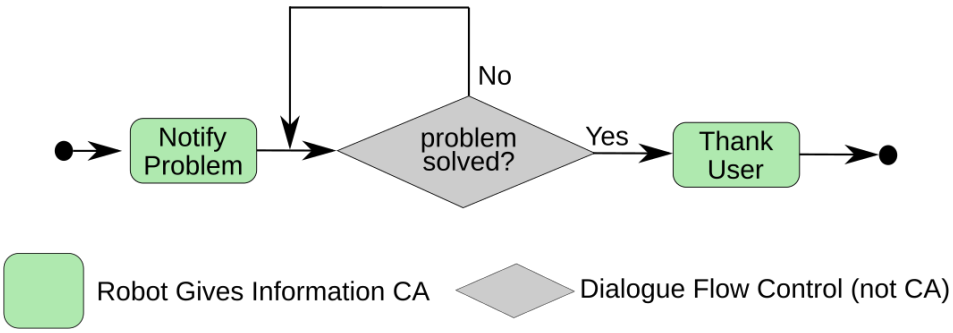


Figure 3.19: Activity diagram for the communication warning CCA. This CCA is requested if external peripherals used by the robot stop working. The CCA requests help from the user, waits for the problem to be solved and then thanks the user.

This CCA, shown in Figure 3.19, uses a *Robot Gives Information* CA to notify the user about the problem with the communication channel and to provide the instructions required to solve it, if possible, or to instruct him/her to seek for help, if the problem cannot be fixed by the user. On top of notifying the user, the CCA also warns the applications about the communication problem, so they do not try to use the broken interface to interact with the user. Once the warning has been issued, the CCA waits until the connection between the HRI Manager and the broken interface is fixed. Finally, the CCA thanks the user for fixing the issue, and then finishes its execution. One of the more common examples of this CCA is related to the touch screen. The connection between the screen and the robot tends to fail from time to time. When this happens, the *Communication Warning* CCA is used to ask the user to close the application running in the tablet and launch it again.

3.5.7 Handling Errors in Communication with Basic and Complex CAs

Uncertainties in interactions are so common that all CAs should be equipped with a set of basic features to deal with them. One of the uncertainties already discussed in this section is the possibility of a failure of the robot's input sensors, either due to the dynamic characteristics of the environment itself (for example, trying to retrieve the voice of the user in a very noisy environment, or the user not speaking in a clear enough manner) or due to an internal problem with the hardware and/or software of the robot. Under this situation, the Perception Manager would send a notification about the failure of the recognition process (Currently, the only communication problems considered are related to speech-based interactions, as the other communication channels have shown to be more reliable). To solve this problem, CAs are equipped with a strategy that allows to ask the user to repeat

his/her last utterance if a recognition failure is received. If the issue is that an input has been received, but the confidence level is not high enough, the CA will inform the user about what has been recognized, and then asks him/her to repeat the utterance anyway. In this case, the CA also stores the confidence level associated to that particular recognition. If the same utterance is recognized, the CA adds the new confidence level to the stored value. This allows the CA to make the decision that a low confidence recognition is correct if it has been recognized multiple times in a row. Regardless of the issue, if the communication problem persists, CAs will give up after three failed attempts and just notify the application that the interaction has been cancelled due to communication issues.

A second uncertainty that has been identified as being fairly common is also connected to the process of retrieving information from the user. Specifically, the problem comes when the user decides to ignore the robot, or just stops interacting without notifying the robot. In this situation, if the robot keeps waiting for an answer, or just keeps repeating the request for information, it could lead to the system getting stuck. To avoid this, CAs can define the amount of time that the user has to provide an input (this is one of the CA-specific parameters that can be configured by the application in the activation request). If the user does not deliver an input in time, then the CA will try to encourage the user to give the information, clarifying the input channel that should be used. For example, in menu-based interactions, the robot would ask the user to select one of the options displayed in the touch screen. After three failed attempts to obtain an answer, the CA cancels the interaction and notifies the application that the engagement with the user has been lost.

In addition to the embedded recovery mechanisms, three CCAs have been developed to deal with communication issues. These CCAs are the *Question with Confirmation*, *Switching Mode Question*, and *Communication Warning CCA*.

3.6 Evaluation of the proposed Dialogue Manager

This section presents the results of the implementation and integration of the HRI Manager in the robot's architecture. First, a case of use is presented in order to show the operation of the proposed dialogue manager in a real environment (Section 3.6.1) and to illustrate the particularities of the system's implementation. Because the control over interactions is divided between the applications and the HRI Manager, the perception that users have of the robot during an interaction is caused by both the application design and the features the proposed dialogue architecture introduces. Thus, it would be tough to extract from a subjective evaluation conclusions related exclusively to the HRI

Manager. Next, an objective evaluation is conducted in order to evaluate the performance of the system (Section 3.6.2). This includes measurements of the resources used (both CPU and RAM usage), and also statistics about the performance of recovery methods in interactions with real users.

3.6.1 Subjective evaluation: Case of use

In the proposed case of use, the robot Mini interacts with older adults in the context of a series of cognitive stimulation exercises (one of the main features included in the robotic platforms used in this thesis). These tests were conducted in a daycare centre located in Zamora (Spain). Figure 3.20 depicts an example of an interaction between Mini and an participant. The presented case of use is a compilation of situations that arose during the trials, and that can be used as a showcase of the features of the proposed dialogue manager. Medical personnel working in the daycare centre reviewed and approved the experiments conducted, and participants gave their signed consent when asked to participate in the trials.

Figure 3.20: Example of an interaction between Mini and one of the participants in the trials.

In these trials, the robot Mini conducted a cognitive stimulation therapy composed of a sequence of exercises that test different abilities of the user (memory, perception, etc...). The goal is to stimulate

the user's brain and try to slow down the deterioration of his/her cognitive abilities. The robot was placed on top of a table in an office, and set in a standby state that imitates the appearance of being asleep. The participant sat in front of the robot, accompanied by a member of the facility's medical personnel and one of the robot's developers. After introducing he robot to the participant and explain how the trial was going to work, the participant woke up the robot by touching its shoulder. After performing an *waking up* expression, the robot greeted the participant and asked him/her what to do next. A menu displayed in the robot's touch screen allowed to select one of the robot's applications. The participant selected the $App_{exercise}$ and, after acknowledging the selection, the robot started the session.

There is a second application that is relevant for the presented case of use: $App_{warnings}$. The robot maintains a schedule where the user can program warnings and notifications that will be issued by the robot to help him/her remember certain events. For example, given the robot's target audience, the robot could remind the user about his/her medicines, or about an incoming visit from a relative. Due to the fact that these warnings are tied to specific moments in time, the $App_{warnings}$ will send its CAs always with high priority, so they can interrupt what the robot is doing at any time. The $App_{exercise}$, on the other hand, is a regular application, and thus will have medium priority.

The following sections present 5 particular situations that arose during the trials conducted with Mini. These situations demonstrate how the proposed dialogue system can be used to create human-robot interactions as combinations of basic and complex CAs. They also serve as a showcase for most of the features provided by the CAs and the HRI Manager.

3.6.1.1 Interaction 1: waking up Mini

The first situation described took place at the beginning of one of the trials, and involves the participant waking up the robot, selecting the $App_{exercise}$ and solving the first question in the first cognitive stimulation exercise. It is an example of how a standard interaction is conducted using CAs. In this context, a standard interaction can be defined as a dialogue between the user and the robot where nothing unexpected happens.

Whenever the robot goes into the *sleeping state*, the DMS requests the activation of a *User Gives Information* CA configured to receive information coming from either the touch sensors (specifically, the right shoulder) or the ASR. These are the channels that can be used to wake up the robot. In the interaction, after the participant woke up the robot by rubbing its shoulder, the robot

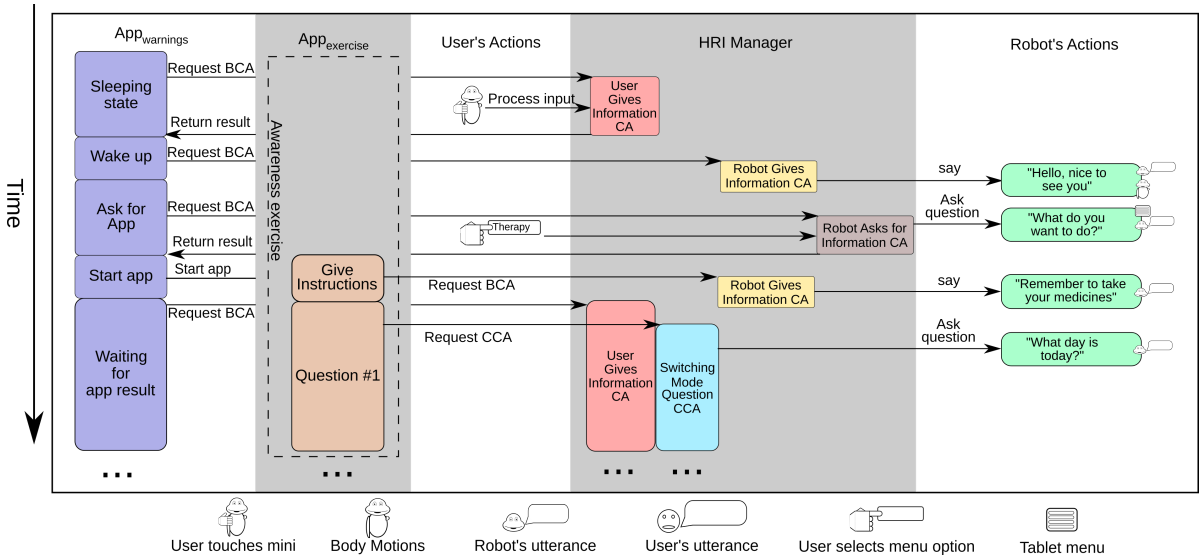


Figure 3.21: Initial interaction between Mini and the user, from the moment the user wakes the robot until the first question in the exercise is asked.

responded by using a *Robot Gives Information* CA to perform a *waking up* expression and greet the participant, and finally asked the participant what she wanted to do. This was done using a *Robot Asks for Information* CA that sends a menu to the touch screen and allows the participant to provide an answer either through voice commands or by selecting one of the options in the menu. When the participant selected the *exercise* option in the menu, the DMS received this answer and launched the $App_{exercise}$. In turn, the app started by requesting the activation of an *User Gives Information* CA that will be used to capture any command that the participant issues to control the application. All applications can be paused, stopped, and resumed at any time by touching the robot's shoulder. This CA will be configured as *continuous*, so it can capture every command coming from the participant, and will remain active until the application itself is stopped. At the same, the app explained how the cognitive stimulation session is going to work and the rules of the first exercise using a *Robot Gives Information* CA. This exercise is designed to test the awareness of the user through a series of general questions, like *What day are we in?* or *What month are we in?*.

In order to shield the interaction from communication problems, all the questions in the exercise are asked using the *Switching Mode* CCA. As shown in Section 3.5.6, this complex CA can be configured to try to retrieve the answer from the user through a sequence of interfaces, so if one fails, the CCA just moves to the next one. In the $App_{exercise}$, the CCAs are configured to use speech-based answers as the default mode, and then switch to tablet-based answers if speech communication proves impossible. The first question asked was *What season are we in?*. The trials were conducted

in February, so the right answer was *winter*. Once the participant provided the correct answer, the CCA sent an expression to the Expression Manager for congratulating the participant. This was done by uttering a congratulation message while the robot shakes its arms up and down, changes the expression of the eyes to happy, and also changes the colour of the heart. Then, the CCA informed the $App_{exercise}$ that the interaction had been a success, and the application continued with the remaining questions in the exercise.

3.6.1.2 Interaction 2: communication problems

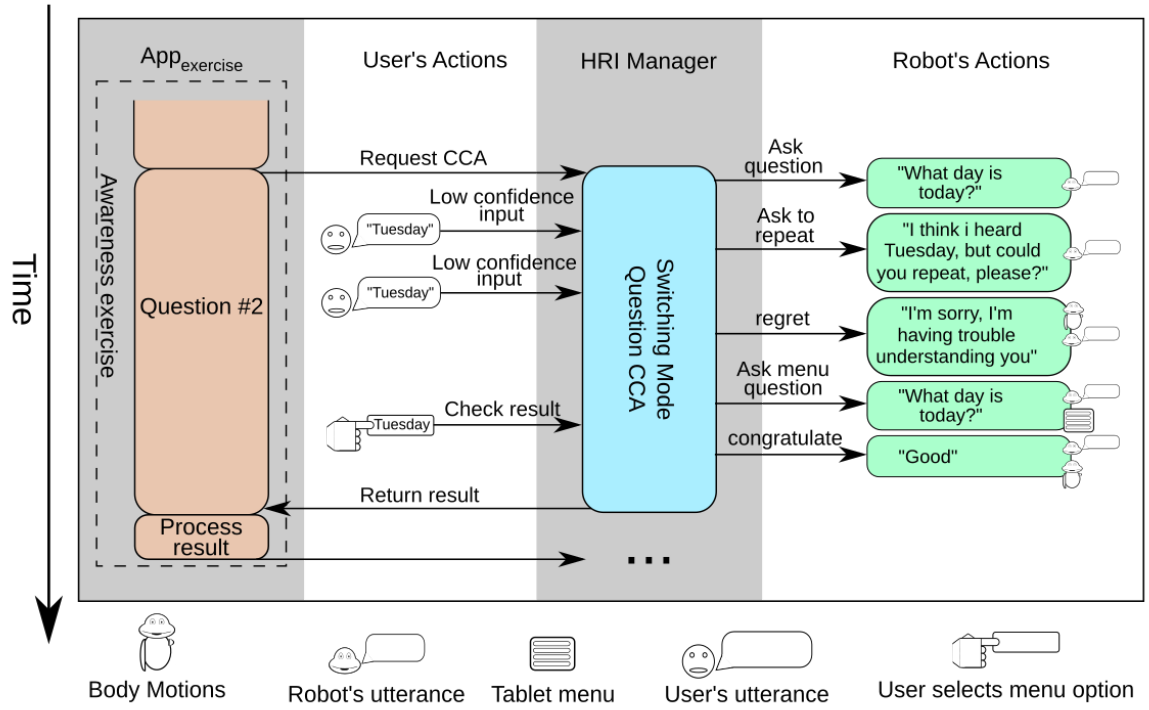


Figure 3.22: Second question of the awareness exercise. The boxes represent actions performed by the different agents and the arrows represent the connections between these actions.

The second situation that will be highlighted in this review, shown in Figure 3.22, happened in the same exercise (awareness evaluation), and has to do with the presence of unexpected conditions that hinder the communication between the robot and the user, and the basic strategies that CAs can use to manage them. In the awareness exercise, the second question was *What day is today?* (in this case, the question was regarding to the month, not the week). Again, the CA requested was the *Sequential Mode Question* CCA, configured so the robot asks the question aloud, and then waits for a speech-based answer. In one of the trials, the participant did know the answer,

and tried to say it a couple of times. Although the ASR was able to generate a recognition, this would not have a confidence high enough to be accepted by the CA (the threshold for accepting or rejecting a recognition is 0.4). The mechanism implemented in the CAs for managing this particular situation is to repeat to the participant the result provided by the ASR and then ask the participant to repeat his/her answer so the robot can get a clearer recognition. In the situation presented here, the participant kept answering, trying to be clearer, but the ASR kept failing at providing a confident recognition. After the third failed attempt to retrieve the participant's answer, the CA switched from a speech-based to a menu-based answer, and repeated the question to the participant, but this time displaying a menu in the touch screen. This solved the communication problems, and allowed the participant to provide the correct answer. This lead to the CA congratulating the participant with the procedure described at the end of Interaction 1, and then informing the $App_{exercise}$ that the interaction was a success (the result sent to the application also includes the amount of attempts that failed due to communication problems). The application then moved on to the next question.

3.6.1.3 Interaction 3: failing to answer in time

The next relevant interaction that deserves to be highlighted, displayed in Figure 3.23, happened while the participant was going through a different exercise. It showcases another of the generic mechanisms that CA have integrated for dealing with unexpected situations (in particular when the user fails to provide an answer or selects the wrong one). In this exercise, the user is presented with a picture of a famous monument and had to answer in which city it is located, selecting one of the options in a menu displayed in the touch screen. The goal of the exercise is to test the user's executive function (the cognitive processes and mental skills oriented to planning, monitoring, and executing goals).

The first question in the exercise had the robot showing a picture of the Leaning Tower, located in Pisa (Italy). Again, the $App_{exercise}$ used a *Switching Mode Question* CCA to do this. But in one of the trials conducted, the participant was not able to provide an answer in the amount of time defined in the CA. When this happened, the *Switching Mode Question* CCA tried to encourage the participant to provide an answer, while reminding her that in this exercise she had to use the menu to answer, just in case she forgot. The question was repeated, and the menu was displayed again. The participant kept doubting about the city where the Leaning Tower is located, but because she wanted to answer in time. this led to her selecting a wrong option. The robot then told the participant that the answer was wrong, encouraged her to try again, and finally repeated the question. Again, the option selected by the participant was the wrong one, and on top of that, she had run out of

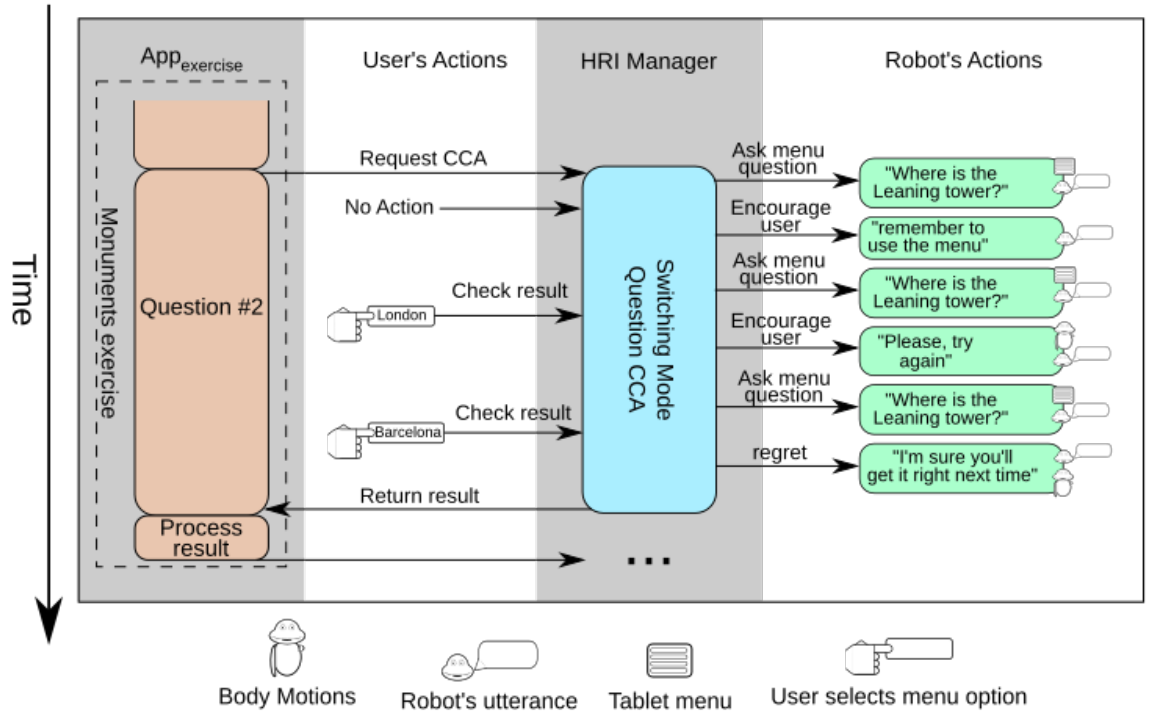


Figure 3.23: First question of the monuments exercise. The boxes represent actions performed by the different agents and the arrows represent the connections between these actions.

attempts. Thus, the CA sent an expression oriented to cheer the participant, using an encouraging utterance, a change on the eyes' expression and the heart's colour. As always, the result of the interaction was sent to the $App_{exercise}$, and the robot continued with the next question in the exercise.

3.6.1.4 Interaction 4: pausing CA with lower priority

The fourth interaction presented here, shown in Figure 3.24, is an example of how the priority system is used in a real situation, and how a high priority CA can interrupt the task that is currently being performed by the robot. As a test, the robot had some warnings already programmed in the internal schedule. In particular, the robot had to notify whenever is time for the user to take his/her medicines. This warning was still active during the trials conducted.

In this interaction, the robot asked the user where the Eiffel Tower is located using one of the *Switching Mode Question* CCA. While the participant was thinking about the answer, the $App_{warnings}$ detected that it was time to trigger one of the preprogramed warnings. Thus, a *Robot Gives Information* CA was requested with high priority so the robot would utter the reminder

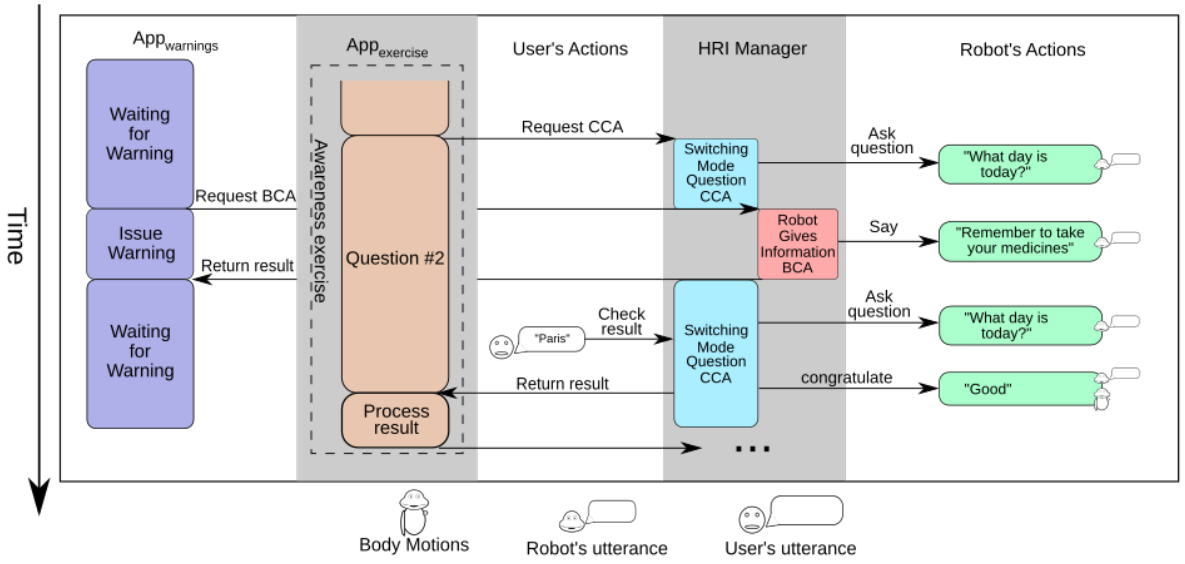


Figure 3.24: Example of a situation with several CAs with different priorities. The boxes represent actions performed by the different agents and the arrows represent the connections between these actions.

for the user to take his/her medicines. Due to the fact that this CA has a higher priority, the HRI Manager cancelled the *Switching Mode Question* CCA, stored it in the appropriate priority queue (medium priority, in this case), and executed the *Robot Gives Information* CA. The robot uttered the reminder about the medicines, extracted the *Switching Mode Question* CCA from the priority queue and executed it again to continue with the exercise. The participant provided the right location for the Eiffel Tower, the CCA congratulated the participant with the expression described in Interaction 1, and returned the result of the interaction to the $App_{exercise}$.

3.6.1.5 Interaction 5: recovering from a network error

The last situation that will be presented in this case of study, depicted in figure 3.25 shows how the system can deal with a complete breakdown of the communication between the HRI Architecture and one of the robot's interfaces. This is a feature provided by one of the CCAs presented in Section 3.5.6: the *Communication Warning* CCA.

This interaction took place while the participant was completing an exercise designed to evaluate his perception skills. In this exercise, the $App_{warnings}$ sends to the tablet a menu with multiple buttons, each of them with an image. The task of the participant is to select the image that is

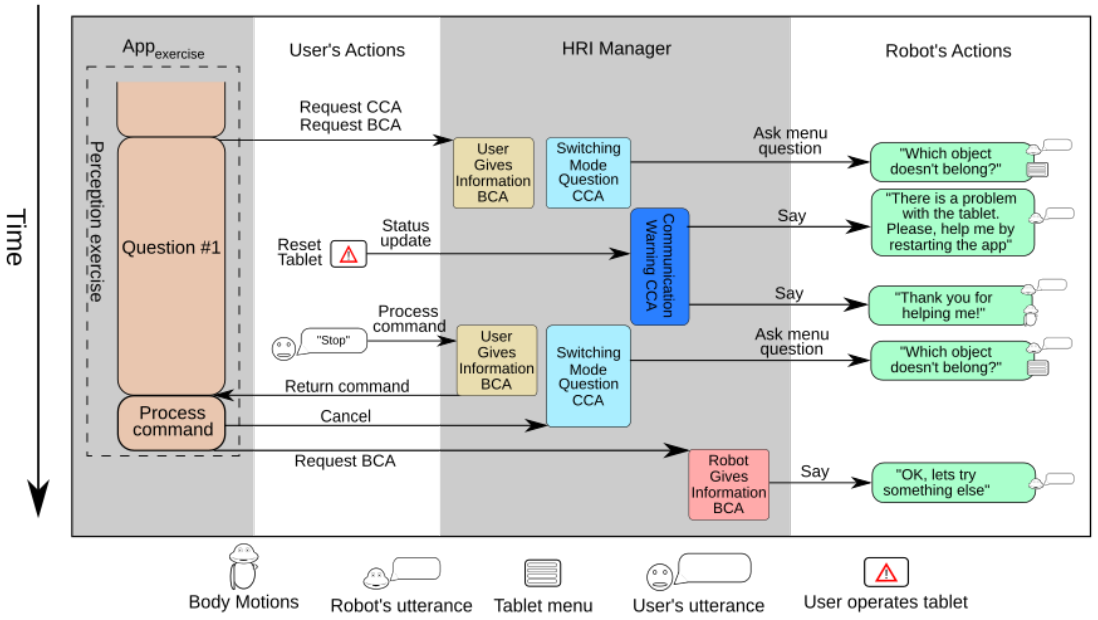


Figure 3.25: Fourth question of the exercise. The boxes represent actions performed by the different agents and the arrows represent the connections between these actions.

different from the others. As in all the other exercises, these questions are asked using *Switching Mode Question* CCA. In the situation presented here, the robot displayed a menu with the following images: a cat, a tiger, a panther, a leopard, and a dog. In this case, all animals are feline except for the dog, which makes it the correct answer. While the participant was considering which option to select, suddenly the connection with the table was interrupted, making the menu disappear.

The proposed interaction architecture has implemented safety measures for this type of problems. In the situation described here, the tablet sends a signal to the robot at a fixed rate. On the other end, the Expression Manager has a watchdog timer that is reset every time the signal is received. If the timer goes off, the Expression Manager starts the recovery process by requesting the activation of the *Communication Warning* CCA with high priority. The HRI Manager cancelled the question that was active, stored it in the corresponding priority queue, and loaded the *Communication Warning* CCA. First, the participant was warned about the problem with the tablet, and asked to reset the application. The developer that was present during the trials restarted the application. When the tablet started to send the signal again, the Expression Manager notified the CA that the problem had been fixed. The *Communication Warning* CCA thanked the user using an expression that shows a display of happiness, changing the expression of the eyes and the colour of the heart of the robot.

Once the emergency was solved, the HRI Manager extracted the *Switching Mode Question* CCA from the priority queue and executed it again. The robot repeated the question, and waited for the participant to answer. At that point, the participant informed the developer that he was already tired and wanted to stop the session. Using the *User Gives Information* CA that was activated at the beginning of the session, the participant touched the robot's shoulder. The DMS paused the $App_{exercise}$ and used an *Robot Asks for Information* CA to ask the participant if he wanted to continue with the session. This explicit confirmation request is used to ensure that an application will not be cancelled by mistake, or by a perception error in the touch sensor. The participant confirmed the cancellation order, prompting the DMS to stop the $App_{warnings}$ and request the cancellation of any CA related to the cognitive stimulation session.

3.6.2 Objective evaluation

This section presents an objective evaluation of the HRI Manager and the CAs aimed at demonstrating that the proposed dialogue modelling and management approach is able to comply with the constraints that exist in a real world application. The objective metrics presented in this section are three: (i) the use of hardware resources (namely, CPU and RAM usage), which shows that the proposed system can be integrated inside a robotic software architecture without the need for a specially powerful hardware; (ii) the response time of the system during different parts of the interactions, which shows that the CAs are able to abide by the temporal constraints that dominate human-human interactions; and (iii) the statistics regarding the use of the different mechanisms integrated in the CAs to manage unexpected situations that arise during interactions, as a demonstration of their effectiveness to handle these situations in a real environment.

3.6.2.1 Resource usage

When developing any robotics application oriented to be integrated in a real platform, success cannot be measured exclusively on the performance of said application. It is not enough that the new module is able to carry its task successfully, it also needs to do it in an efficient manner that minimizes the amount of resources required to run this application. This becomes even more important if the new module will have to perform any of the tasks that can be considered as essential in any social robot. Dialogue management can be considered one of this tasks, at least in the context of this thesis.

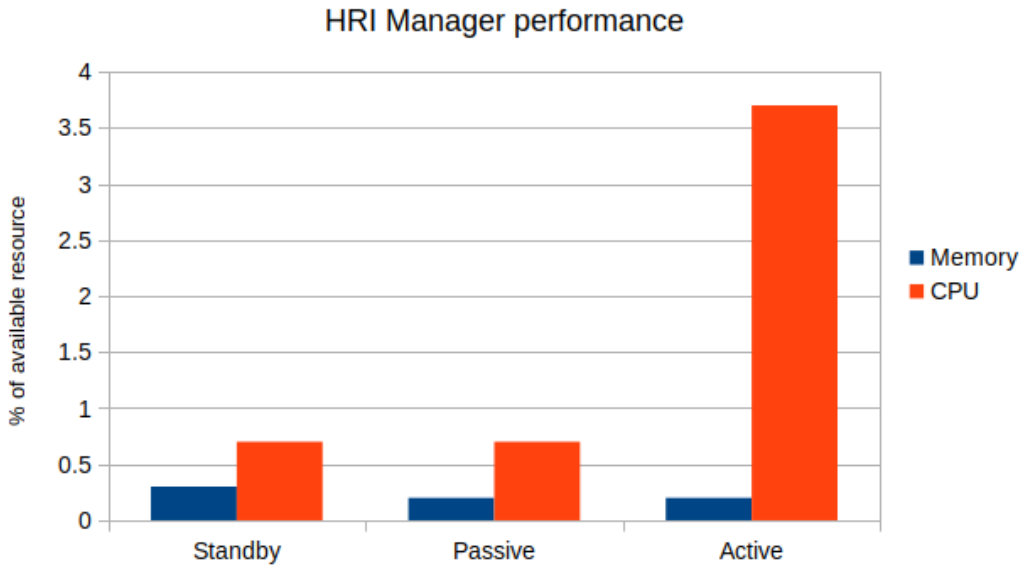


Figure 3.26: Graphics showing the resources used by the HRI Manager. The use of CPU is computed as a percentage of the processing power of a single core.

All the measurements that will be presented in this section have been taken in Mini, under real conditions (all the modules of the robot running at the same time). Mini is equipped with 16 GB of RAM, and an Intel i5-3550 CPU, with four cores running at 3.3 GHz. The software architecture is running on Ubuntu 16.04 64 bits. Measurements were taken under three different conditions: *standby*, *passive*, and *active*.

Under the *standby* condition, the HRI Manager was tested individually, without launching any other module in the robot's architecture. In this situation, the usage of RAM measured was a 0.2% of the available memory, a value that showed no variation through time. Regarding the CPU usage, this variated between a 0.0 and 0.7% of the processing capacity provided by one of the CPU's cores. Under the *passive* condition, the whole software architecture of the robot was launched, and the robot was left in the standby state. In this state, the HRI Manager was maintaining a *User Gives Information* CA active for waking up the robot. The measurements shown the same memory and CPU usage that the ones measured when the HRI Management was working alone. Finally, during the *active* condition, measurements for both resources were taken during an active interaction, where *Robot Gives Information* and *Robot Asking for Information* CAs were used to establish a dialogue with the user, while the *User Gives Information* CA was kept active. Under these conditions, which can be defined as the regular state of the robot whenever is not in the standby state, the memory usage was maintained (0.2% of the available RAM). An increase on CPU usage was perceived, with

fluctuations that went from a 0.0% at the lowest point, and a 3.7% at its highest. Being one of the core systems of the robot, requiring the allocation of $1/500$ of available RAM and $1/108$ of the robot's processing capacity has been considered to be acceptable.

3.6.2.2 Response time

It has been already established in multiple sections of this manuscript that the response time is a feature that plays a big role in perceiving interactions as natural. This is a reality not only in HRI, but also in interactions that occur between humans. As an example, lets consider an interaction between two friends. At some point one of them shows no sign of acknowledgement of the previous communicative action performed by the other. In this situation, the person that has been ignored usually tries to perform a possible action oriented to repair the communication (repeat the sentence, try to get the other person's attention, or ascertain what the problem might be) or even disengage from the conversation.

There is a design rule that is commonly followed when defining the threshold that can be used as the limit for maintaining a natural interaction. This is known as the *"Two Second" rule* [99] and, as it can be inferred from the name, establishes that the response of the system in a human-computer interaction has to be provided in under two seconds to be considered acceptable. Other studies set this threshold in under a second [100]. There is also some research that shows that users tend to prefer robots that delay responses for some time (i.e. 1 second) than robots that give immediate responses [101]. In order to be as restrictive as possible, one second will be the response time limit that will be used for evaluating the performance of the proposed system.

All CAs in the proposed approach have been equipped with loggers that assign timestamps to each of the steps performed during an interaction. These timestamped points in the interaction have been retrieved for all the CAs that were involved in the situations described in the subjective evaluation. In this study, three different situations have been identified as having a potential effect over the perception that the user has of the interactions:

- **Action time:** How long it takes the robot to convey any message that has to be transmitted to the user. This includes the time that passes from the moment the HRI Manager receives the CA activation request until the communicative action that the robot has to perform has been sent to the Expression Manager. The time required for performing said action is left outside of this evaluation, as it is not affected by the implementation of the dialogue manager. If the CA

requested needs to configure any output or input interfaces for achieving its communicative goal, the time required to do it is also included here. Finally, any delay in the execution of a CA as a result of priority management operations (the time lost in stopping the active CA and store it in its priority queue) is also included in this category. All CAs, regardless of the type of communicative goal they model will be affected by this.

- **Reaction time:** How long it takes the robot to process the communicative actions performed by the user (for example, utter a sentence, select an option from a menu displayed in the tablet, or touching the robot) and generate an appropriate response, if required. Not all CAs are designed to expect any actions coming from the user, and some of the ones that do expect such actions do not respond to them, so this time will vary from CA to CA.
- **Completion time:** How long it takes the CA to complete its execution after the interaction modelled by it has been completed. It encompasses the time between the interaction's end until the moment the HRI Manager publishes the result of this interaction. Usually, this time is not directly perceived by the user, but instead it is combined with the first situation described, as the time that passes since one robot action to the next one involves completing the execution of one CA and configuring and executing the next one. All of the CAs developed have to go through this process.

The response time is going to show a big variation not only between different CAs, but also between different configurations of the same CA. This is due to the fact that not all CAs have the same mechanisms integrated. For example, while both the *Robot Asks for Information* CA and the *Question* CCA are used to retrieve information from the user, only the second provides feedback to the user after. There are other factors that will play a role in the response time, like the amount of CAs running in parallel, the existence of CAs with different priorities, or even the communication channel used to retrieve the user's action (e.g. a speech-based answer is more complex to process than a touch-based one). All the measurements extracted for this analysis were taken across five different participants in the trials. The results were extracted for the same CA in the session for all participants, in order to ensure that the particular configuration of the CAs did not have an effect. Then, the average for all five interactions was computed.

Figure 3.27 presents the results obtained for the three situations considered for each type of CA. In this case, bars represent the average value for each type of CA, while the whiskers represent the standard deviation. Figure 3.27 shows that the response time for all CAs is significantly below the selected threshold of one second, which allows to confirm that the proposed dialogue manager is able to abide by the temporal constraints imposed in Human-Robot Interactions, without imposing time

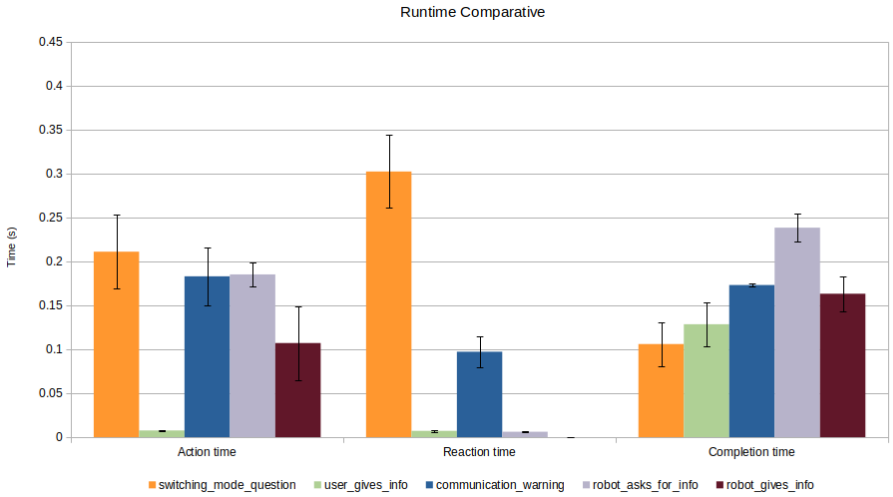


Figure 3.27: Measurements for each of the three times defined (Action time, reaction time, completion time) for each of the CAs used in the subjective evaluation. The bars represent the average value, while the whiskers represent the standard deviation.

limitations on other modules of the architecture (for example, in order to achieve the one second response time, according to the results presented, the Expression Manager would have to perform its task in under 0.65 seconds, which is manageable). Next, the response times measured for each of the five interactions presented above are discussed more in depth.

Figure 3.28 shows an overview of the Action, Reaction, and Completion Time for each of the CAs that are used in the first situation presented during the subjective evaluation. *Greeting* and *Instructions* correspond to instances of the *Robot Gives Information* CA, *Select App* represents the *Robot Asks for Information* CA used to ask the participant want he/she wants to do, *App Control* is the *User Gives Information* CA that allows the participant to control the cognitive stimulation session, and *Question 1* represents the *Switching Mode Question* CCA used by the robot to ask the first question in the exercise. As expected, the *Switching Mode Question* CCA presents a significantly higher Action and Reaction Time delays. The former is due to the fact that, being a complex CA, needs to build the state machines for each of the BCAs that compose it. The latter is because it is the only CA used that provides feedback to the participant. In any case, all the delays measured are under 0.33 seconds, which is well below the threshold established. Is important to mention that the *User Gives Information* CA does not present Reaction or Completion Time because this CA played no role in Interaction 1, besides being activated.

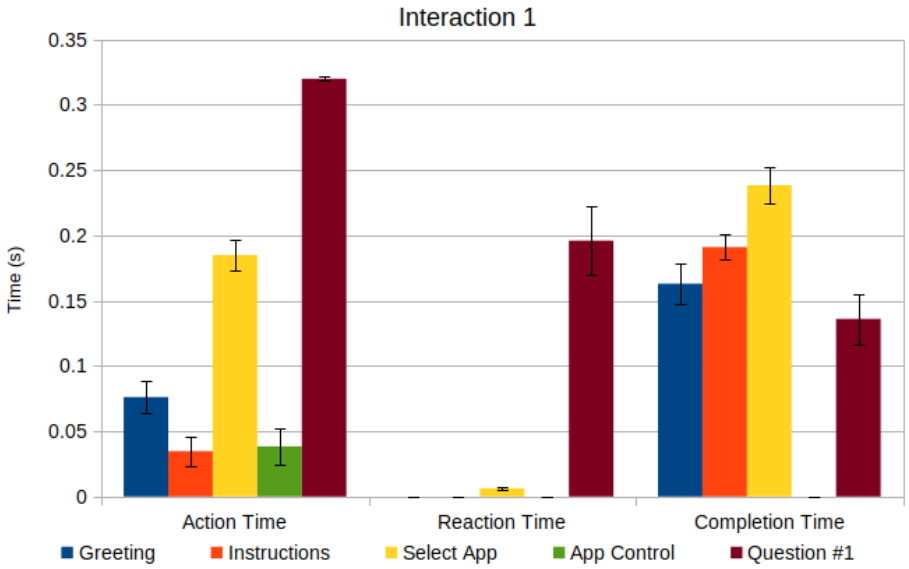


Figure 3.28: Measurements for each of the three times defined (Action time, reaction time, completion time) for each of the CAs used in interaction 1. The bars represent the average value, while the whiskers represent the standard deviation.

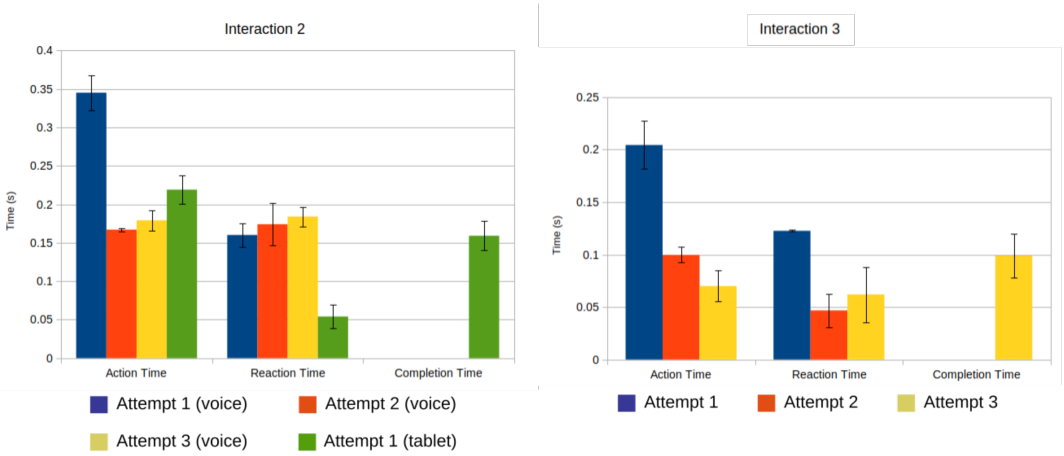


Figure 3.29: Measurements for each of the three times defined (Action time, reaction time, completion time) for each of the CAs used in interactions 2 and 3. The bars represent the average value, while the whiskers represent the standard deviation.

The detailed measurements taken for the CAs used in Interactions 2 and 3 are shown in Figure 3.29, respectively. Both situations can be analysed together, as in both cases the CCA attempts to retrieve a speech-based answer three times (after the third time, the result of the interaction is sent back to the application in Interaction 3, while in Interaction 2 the CCA switches the input mode to the tablet). Regarding the Action Time, it is significantly higher for the first attempt than for

any of the others in both Interaction 2 and 3. This is due to the fact that the Action Time for the first attempt includes the time required to build and configure the CA, while for the others it only includes the time required to sent the question to the Expression Manager. The fact that the second highest Action Time is tied to the attempt that uses the tablet in Interaction 2 was also expected, as it includes the delay introduced by the process of switching interfaces. For the Reaction Time, the fact that it is higher in all the attempts that use the speech as input method could be explained due to the higher complexity involved in processing this type of answers. However, there is no particular explanation for the variations in Reaction Time that can be observed in Interaction 3 between the first attempt and the other two. Similar to the results obtained for Interaction 1, the highest delay observed in both Interactions 2 and 3 is the time required for setting up the CCA.

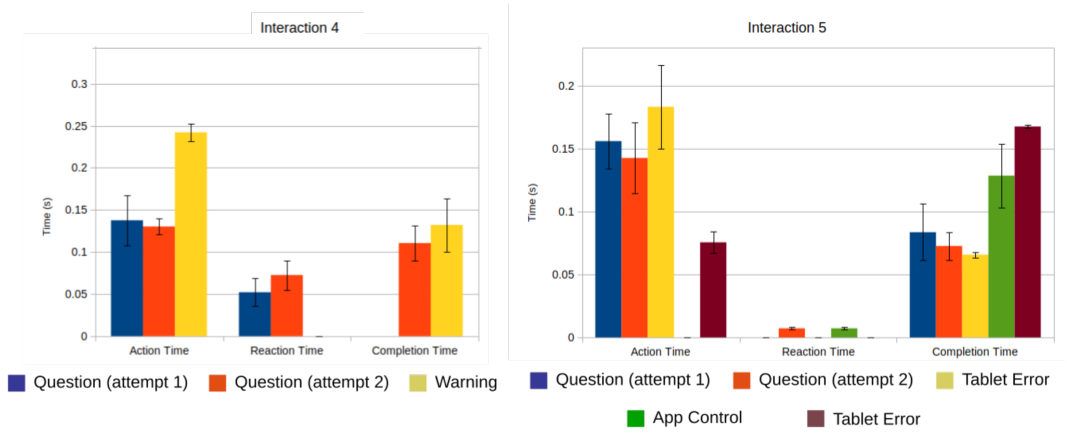


Figure 3.30: Measurements for each of the three times defined (Action time, reaction time, completion time) for each of the CAs used in interactions 4 and 5. The bars represent the average value, while the whiskers represent the standard deviation.

Like in the previous case, Interactions 4 and 5 present a similarity that can be exploited for comparing them. In both situations, the task that the robot was performing had to be interrupted to manage a CA with a higher priority. The results are shown in Figure 3.30. As in all of the previous interactions, the highest delay is the Action Time, in this case the one related to the CA with high priority. This was expected, as the Action Time includes the time required for cancelling the CA with lower priority. Overall, the detailed evaluation of all the interactions presented in the subjective evaluation seems to confirm the idea that the proposed CAs are able to work under the time constraints that have been defined for a natural Human-Robot Interaction.

3.6.2.3 Interaction statistics

The last evaluation performed is an overall study of how the interactions between the robot and the participants in the trials were conducted, analysing the appearance of problems that triggered the intervention of any of the mechanisms integrated in the CAs, and what percentage of these interventions where successful. As part of the experiments conducted, all interactions between the participants and the robot were recorded. This analysis was performed over four of these trials, all conducted by different participants, and in different days. The overall duration of all interactions reviewed is 71 minutes, combining the results from all participants. In this time, 354 CAs were performed, including complex CAs. This amount of CAs includes only CAs where the robot has the initiative, as the *User Gives Information* CA cannot be perceived in the videos unless the user makes use of them, which did not happened in the interactions analysed. In any case, this is not a big drawback of the study, as the CAs for user-initiated interactions that were used in these trials tend to be the most robust, due to the fact that they generally use the robot's touch sensors to retrieve information. In the videos reviewed, only 41 out of the 354 CAs used faced any complication and had to resort to any of the built-in recovery mechanisms. This means that an 88.42% of the interactions were conducted without any complication by means of the HRI architecture presented in this thesis. A summary of this information is presented in Figure 3.31.

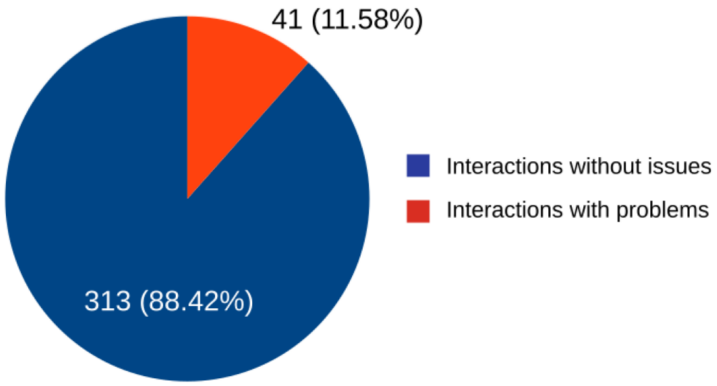


Figure 3.31: Summary of the interactions reviewed.

During the 41 interactions that faced any communicative problem, CAs had to activate one or more recovery mechanisms a total of 65 times. This means that some of the CAs had to face more than one situation where the interaction was interrupted by one or other issue. Out of these 65 situations handled by the CAs, 29 of them occurred during (and due to the use of) speech-based interactions. This represents a 44.62% of all uses of recovery strategies. Usually, voice-related communication problems involve the ASR failing to provide an accurate recognition of the

participant's speech, either due to noise in the environment, unclear voice, or problems with the robot's microphone. Specifically, in 17 of these 29 speech-related problems, the ASR was not able to recognize any speech, while in the other 12 a recognition was generated, but with a confidence level below the threshold required by the CAs. The second most common situation that arose during the interactions evaluated, accounting for 19 out of the 65 situations handled (29.23% of the total), involved the user not providing an answer in time. This includes interactions where the participant did not know what to answer (inferred from the conversation between the participant and the personnel present during the trials), took too long to answer, or the microphone failed to record the participant's voice. The remaining 17 times (26.15% of the total) the CAs had to correct the interaction where due to the participant failing to answer one of the questions asked by the robot correctly. This can be handled by inviting the person to submit a new answer (if he/she has any attempt left), or by cheering him/her up (if there are no more attempts left). Figure 3.32 shows a summary of all the situations handled by the built-in recovery mechanisms integrated in the CAs.

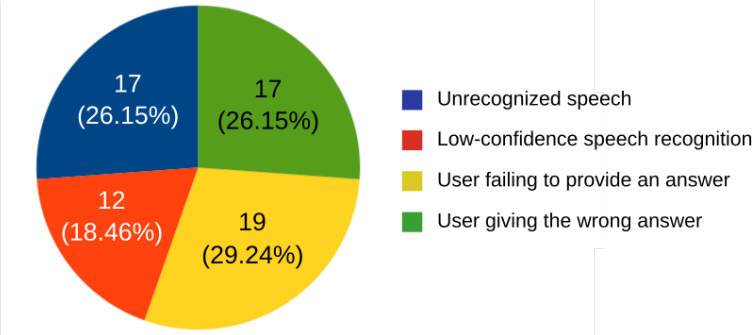


Figure 3.32: Summary of the unexpected situations that had to be managed by the CAs.

The results presented above show that the proposed approach to dialogue management is not only able to manage the majority of dialogue actions in the trials without encountering difficulties, but was also able to manage appropriately those interactions where unexpected problems arose in a way that led to no situations where the communication between the robot and the participant was broken beyond repair. This suggests that the recovery mechanisms that have been integrated in the CAs are able to handle the most common causes of communication failure, although this does not ensure that no more strategies will have to be developed in the future.

3.7 Conclusions

Dialogue Systems are software modules that are used to communicate with humans in an understandable way. This implies using a communication system based on symbols that have an agreed upon meaning (for example, spoken or written language, signs, etc..). Dialogue Systems have to manage a variety of subtasks related with interactions, including managing inputs perceived from the environment, processing them in order to extract the relevant information, or form the communicative actions that the robot will perform during the interactions. The central task that has to be handled by these systems is the control of the interaction's flow, using communicative actions and the perceived inputs to advance the conversation appropriately. This task is usually going to be handled by a core module: the Dialogue Manager. In the work presented in this chapter, the role of the dialogue management is performed by the HRI Manager.

While the proposed system has proven to be useful under real conditions, with the robot interacting with real users (not only developers), the current state is not considered a final version, but instead a stepping stone over which new modelling and management techniques can be integrated and tested, in order to create the best possible experience for the users, and the simplest possible development process for roboticists.

3.7.1 Contributions and achievements

The main goal of this part of the thesis is twofold. First, a dialogue model for Human-Robot Interaction that is based on atomic interaction units has been proposed. These pieces have been named **Communicative Acts**, or CAs. They model small, self-contained dialogues based on two dimensions: intention and initiative. The former represents the communicative goal that has to be achieved with the CA, while the latter represents which speaker is the one imposing his/her goals during the dialogue. In an attempt to make the system highly modular and cover any possible situation that each of the robot's apps might need to handle, the possible communicative goals that can be achieved have been generalized to the maximum degree possible: either the speaker with the initiative tries to convey information to the other peer, or tries to retrieve information from him/her. Any other goal normally identified in interactions can be considered as a particular case of one of these two generic intentions. For example, encourage, warn, explain, or greet could be considered as communicative goals on their own, but all of them involve conveying certain information to the other peer, with the specific objective being defined by the type of information. Regarding the initiative dimension, only two possibilities were considered: either the robot has the initiative or the

user does. This is because the dialogue model was developed for robotic platforms designed with strictly one-to-one interactions in mind.

The second major contribution presented in this chapter is the implementation of the HRI Manager, the module that processes the requests for interaction sent by the applications and creates the corresponding dialogues as a parametrized combination of communicative acts. It also provides a priority-based conflict management strategy that allows applications to request CAs in an asynchronous way without having to worry about what other applications are doing. Under the proposed implementation, CAs are modelled using structures similar to state machines, and can be loaded and configured to suit the goals of the applications that request them.

The proposed dialogue management structure divides the control of the interactions in two levels: the CAs take care of the tasks that are interaction-specific, while the applications control every aspect that requires task-related information. On top of that, the CAs also integrate a series of mechanisms to control unexpected situations that might arise during a conversation. This includes input perception errors, or user disengagement in the middle of an interaction. This solution has the advantage of giving developers complete freedom to design interactions as better suit their applications, without having to manually design those aspects that are common to all dialogues. The use of generic Communicative Acts that can be parametrized simplifies the development of new dialogues, as the request of all CAs is done through a standardised template that can be easily filled to achieve different goals.

The CAs developed not only can be combined to form complex dialogues, but also can be nested into what has been called **Complex Communicative Acts**. This results in modular interactions where any section of a dialogue can be transformed into a new block that will be used in the future to build interactions. The combination of modularity and parametrisability results in a dialogue manager that takes part of the load of developing new interactions away from the developers, without enforcing limitations on how to structure the dialogues, as long as the structure selected can be modelled as a combination of CAs.

Several tests have been conducted to evaluate if the proposed approach is able to perform under the constraints that human communication imposes. The overall integration of the HRI Manager has been demonstrated through a case of use based on real trials conducted using Mini in a daycare centre, where the robot conducted a cognitive stimulation session with the participants. Some key situations were extracted from these trials to provide an overall example of all the possibilities that

the CAs provide. Also, the results obtained show that the CAs' response time allows the robot to adhere to the rhythm of human interactions, without imposing strict time limitations to the rest of the modules in the architecture. Also, along with the case of use and the temporal analysis, statistics about the interactions were used as an objective measure of the system's performance. It shows that the majority of the interactions were conducted without any problem, and the few problems that appeared were successfully handled by the recovery mechanisms integrated in the CAs.

3.7.2 Achievement of the proposed goals

In Section 3.1.1, the goals that this chapter of the dissertation aspired to achieve were presented. Two main objectives were devised, which in turn were divided into a series of subgoals that would lead to the achievement of the principal goals. This section describes at which degree each of these subgoals have been achieved:

- The first subobjective stated that the dialogue model developed had to allow the use of multimodal communication, in a way that allowed to create any combination of input and output communication channels. This goal has been successfully achieved, through the configuration of the proposed Communicative Acts. The template used to request the activation of CAs allows to specify both the information that will be conveyed to the user, as well as the input channels that can be used to communicate with the robot in the context of the interaction being described by the CA. The applications can define the message that will be conveyed through each different channel individually, or request one of the multimodal expressions that are stored in the gesture library of the robot. Regarding input processing, CAs that have the goal of retrieving information from the user can specify which input channel to use. CAs can force users to communicate through a specific channel, request a message that comes through multiple channels at once, or give the user the freedom to select which channel to use (from a list of options included in the CA's activation request).

- The second subobjective sought to achieve a modular interaction where complex dialogue structures are created as a combination of more basic units. This goal was achieved with the creation of the Communicative Acts, and then the result was reinforced with the development of Complex and Frequent CAs. Under the proposed dialogue model, applications can create interactions as a sequence of BCAs, each of them with a specific communicative objective. The transition between BCAs is defined in the application, following whichever paradigm the developer considered most appropriate. Also, if some dialogue structures are common inside a particular domain, developers can choose to either model them as a *Complex CA* or as a *Frequent CA*. Complex CAs are more complex to craft, but have the advantage of allowing

to combine CAs (basic or even other CCAs) with other SMACH-based states that provide extra functionalities. *Frequent CAs*, on the other hand, are simply a way for defining a small dialogue as a combination of CAs by specifying the configuration of each individual CA, and the transitions between them. They are easier to code, but only provide the functionalities included in the sub-CAs that compose them.

- The third subobjective defined the temporal requisites that the dialogues generated using the HRI Manager and the CAs. In this case, it was a requisite that the dialogues would follow the same temporal constraints that affect human-human interactions. In Section 3.6.2.2, it was defended that the robot should be able to perform any given action in under 1 second, for an interaction to be considered natural (although other works propose different thresholds, like the one extracted from the *"Two second" rule*). The experiments conducted for evaluating the performance of the HRI Manager and the CAs showed that, under the worst case scenario observed in a real situation, the response time of the proposed system was around 0.34 seconds, which is well below the 1 second threshold considered. Based on these results, this goal can be defined as successful.

- The fourth subgoal is connected to the management of communication problems that arise due to perception errors. The goal aimed at developing strategies that ensure that the communication does not break due to these errors, and that the system can manage them correctly. This goal was achieved through the implementation of a safety measure added to all BCAs that try to obtain information from the user. This strategy allows CAs to respond to perception errors or to input information that has been captured with a low confidence level. If no input is perceived, the CA can inform the user of the problem, and then ask him/her to repeat what he/she said or did. If is a confidence issue, the robot will inform the user about the input received. If the same low confidence input keeps being received, the CA can decide to accept it. In any case, if communication proves to be impossible, the CA notifies the application of the failure and ends its execution. The robustness of the proposed dialogue management approach against input recognition errors has been increased with the development of two CCAs that implement more proactive strategies to solve this issue. The first one can ask the user for an explicit confirmation if a low confidence input is received, instead of simply asking him/her to repeat it. The second one allows to define multiple channels for retrieving the input from the user, and switch between them depending on their success to obtain this information. Also, it has to be mentioned that all CAs have the possibility of giving the user the possibility to provide inputs through multiple modes, which could also be used to counteract possible environmental effects that affect a particular input interface (this is not considered

as an strategy for handling communication errors because it puts the responsibility on the application requesting the CA).

- The last subobjective seeks to shield the interactions from unexpected situations that are directly caused by the other peer. The situations considered are two: (i) the user forces a change in the initiative of the dialogue; and (ii) the user disengages and leaves mid-dialogue (or just stops answering). The first situation was solved with the combination of multiple elements. In first place, the initiative was considered as one of the fundamental dimensions in communication, and CAs have been developed to manage interactions where either the robot or the user have the initiative. In second place, the HRI Manager can keep multiple CAs active at once, and indefinitely. A priority management system ensures that there are no conflicts between the active CAs (for example, two CAs that try to use the same interfaces). With this in mind, the HRI Manager allows applications to take the initiative in the dialogue by requesting the activation of CAs for robot-led interactions, and activate multiple user-led CAs for managing potential changes in initiative. Regarding the loss of the other peer during interactions, all CAs have been endowed with watchdog timers that control the amount of time that have passed without receiving a response from the user. If the timer is set off, the CA will at first try to encourage the user to answer. If this fails, the CA will notify the application about the communication failure and ends its execution.

3.7.3 Limitations of the system and future lines of work

After analysing the goals defined for this chapter of the thesis, it can be concluded that the proposed approach fulfils all the needs identified for a dialogue manager that will have to be installed in a social robot. However, there are several aspects that will be addressed by future works:

- In the current implementation of the HRI System presented in Chapter 2, interactions are divided in two levels, where the developers of applications have to design the flow of interactions as a combination of CAs and CCAs. The advantages of this approach have been described in several points of this dissertation. However, developing a method for creating this combination of CAs automatically (while still allowing for a manual combination if desired) could improve the flexibility of the proposed approach to dialogue management. This method would receive the high-level communicative goal that has to be achieved (for example, greet the user), and executes as many CAs as required to achieve this goal. The applications would still have to provide all the task-related information required for achieving this goal.

- The CAs are configured with the channels that can be used to receive information from the user. Although developers have the possibility of specifying multiple interfaces to increase the options that users have, only one CA can use a particular interface (in the case of the voice, multiple CAs can expect this type of information if they use different grammars). This limits the amount of CAs that can be active at the same time, as for example only one CA that uses touch sensors, or a specific grammar for the ASR can be used. Future works should aim at improving the analysis of input information, in order to optimize how this information is passed to the CAs. An extra advantage of this improved analysis is that specifying the interface/s that has to be used to retrieve information would be no longer necessary, and the users would only be limited by the robot's perception capabilities. For example, for a yes/no question, the user could answer verbally, but also select the option in a menu, or even just give a thumbs up or a head shake, and the CA would be able to understand all these actions as responses to the same question.

- Besides forcing developers to specify the communication channels that can be used, there are other limitations regarding the use of input interfaces. For example, for communicating verbally with users, the CAs rely exclusively on a grammar-based ASR for recognizing the user's speech. This implies that the CAs will only be able to understand those utterances that have been taken into account during the creation of the grammars. This limits the options that the user has to respond to the robot, as the grammars might not account for synonyms, or for speech patterns that are different than those from the developers (for example, slang specific to certain cities or areas). Changing to an open ASR and adding a Natural Language Processing unit that can be used to extract all the relevant information from the speech that the CA might need could increment the versatility of the system and reduce the workload for application developers, removing the need for creating new grammars. These improvements might not fall under the HRI Manager responsibilities, but instead have to be implemented in the robot's perception architecture, while the HRI Manager would be in charge of supplying the task-related information that the input modules might require.

- Both Mini and Gero (the robotic platforms in which the proposed dialogue management architecture has been integrated) have been designed with one-to-one interactions in mind. Although the CAs and the HRI Manager have been developed with the idea of being generic tools that can be integrated in any platform, the truth is that their design has focused on this particular type of interactions, from the possible values for the *initiative* dimension considered (only two: robot or user), to the strategy developed for handling changes in initiative (having user-specific CAs active to manage possible situations where the user wants to take control).

An analysis of the requirements that multiparty interactions have should be conducted in order to evaluate if the current functionalities are enough to handle this type of communication, if new CAs are required, or if any of the functionalities implemented needs to be expanded.

- In line with the previous point, there are some limitations that the proposed approach has regarding the management of user-initiated dialogues. The current version of the system relies to a high degree on the applications for this. At any given time, the applications decide in which situations the user can take the initiative and activate user-oriented CAs that have to be constantly running to manage those situations. This means that, in order to allow the user to take the initiative in order to direct the flow of the interaction towards his/her own goals, particular user's initiative-oriented CAs have to be active. For example, while the participants in the case of use presented in Section 3.6 can stop the interaction using the *User Gives Information* CA that is running alongside the CAs for managing the cognitive stimulation exercises, they cannot directly request the robot to switch to a new application, for example. Thus, it would be interesting to expand the capabilities of the HRI Manager to react to user inputs that do not belong to any active CA. A possibility would be the addition of a specific module for controlling these situations. For example, a chatbot could be used to generate appropriate responses to user utterances that do not belong to any CA.

- The division of responsibilities regarding the creation of dialogues that has been implemented under the proposed approach means that every new application that is added to the robot has to implement an algorithm for managing the flow of any possible interaction that the application might need. If the goal is to achieve dynamic conversations that are flexible and feel natural, the design process can have a high complexity. On the other hand, the dialogue management approaches that are easier to implement usually result in rigid interactions that might end up feeling repetitive and mechanical. The HRI Manager provides the *Frequent CAs* as a method for creating short, basic dialogues following a state-based approach, where each CA represents one of the states in the dialogue. Future research should aim at extending this functionality with the addition of other dialogue management paradigms that allow to create more complex interactions without increasing the complexity of creating new dialogues.

- The modelling of the interactions HRI Manager has been designed to ease the development of human-robot interactions, removing from the design process the common functionalities that should be added to every dialogue. Thus, future tests should try to assess the usability of the CAs from the point of view of the developers. These tests should measure the complexity of creating a dialogue as a combination of CAs, or the learning curve required to use the proposed system or to create new CAs, for example.

CHAPTER 4

Designing the expressiveness of a social robot

4.1 Introduction

The design space of a system can be understood as the factors that can be controlled by the designers in order to alter the appearance and behaviour of the system. When focusing on socially interactive robots, this design space involves three elements [102]: (i) contextual factors, (ii) embodiment design, (iii) and behavioural design. Regarding the context of the interaction, two core factors have to be considered: *task* and *social role*. The task is the factor that is going to drive the interactions with the robot. Communication with the social robot will be oriented to fulfilling this task. On the other hand, the social role indicates the role that the robot will play in the interaction. According to [102], the response that will be elicited on humans is going to be different if the robot plays the role of a superior, a subordinate, or a peer.

Second, embodiment is a characteristic that presents a high variability among designs, due to the fact of having multiple features involved. The review presented by Deng et al. [102] focused on two specific dimensions: design metaphor and level of abstraction. The design metaphor indicates the inspiration of the robot's design. For example, the embodiment of the robot can be based on the human body, or take inspiration from a particular animal. The level of abstraction indicates how similar is the embodiment to the design metaphor. This will have an effect on the expectations that the robot will create on people.

The last dimension fo the design space considered by Deng et al. is behaviour design. In regards to the expressiveness aspect of robotics, this involves designing the expressions that the robot will perform. In this chapter, the terms *expression* and *gesture* are used indistinctively to describe a combination of multimodal actions with a specific communicative goal. For example, a *greeting* expression could be represented by the combination of arm motion resulting in the waving of the hand and a greeting utterance, like *"Hello user, nice to see you"*. This chapter will focus on this feature, as both the contextual factors and the embodiment are already determined by the robotic platforms used in this thesis.

4.1.1 Modelling the expressiveness of a social robot

Designing the expressiveness capabilities of a social robot presents a series of challenges that have to be overcome. One of the first things that has to be considered is the communication channels that will be used during interactions. Traditionally, research on communication has focused on the verbal component of expressiveness. As stated in previous chapters, speech is the main communication channel used by humans in day-to-day situations, due to the ability to describe any mental concept in a way that is understandable without the need for a common ground knowledge between speakers. However, multimodality has been gaining more attention in recent years, as it mimics human communication better than relying exclusively on verbal interactions. [103]. The ability to properly combine modes of interaction during dialogues increases the communicative efficiency of the speaker and adds robustness to the dialogue. Non-verbal communication plays an important role in communication, accounting for 55% of all the information conveyed in a message [104]. The remaining 45% is attributed to speech (7% for words, and 38% for vocal aspects). Phutela [105] defended that body language contributed in the same percentage to the impression that the speaker creates on others.

There have been multiple researches aimed at evaluating the importance that non-verbal communication has. A large part of these works have focused on specific situations and environments, although some authors have tried to give a general vision. For example, Zeki [106] studied the importance that non-verbal communication has on a college classroom environment. The results of this study show that the use of non-verbal cues can improve the student's motivation and concentration, as well as help the teacher to attract and keep attention. Wang [107] studied the effect that non-verbal communication has on interpersonal communication. Four elements were considered in this work: body behaviour, both through motion and posture, and also through appearances; the use of space and interpersonal distance in interactions; silence as a communicative

tool; and signs and symbols. Wang defends that the study of nonverbal communication can lead to an improvement on how we use non-verbal behaviours, thus leading to a more effective interpersonal communication. In 1986, Sabatelli and Rubin [108] presented the results of a study that suggested that people who use more spontaneous non-verbal communication are perceived as being more interpersonally attractive than those who display limited non-verbal expressiveness. In their study, the non-verbal behaviours considered were those that are displayed intentionally. Earlier studies, like those presented by Strong et al. [109] or Bayes [110], had already suggested a relation between non-verbal expressiveness and interpersonal attractiveness. Thus, evidence points towards the importance of developing embodied social robots that are able to use multimodal expressions during interactions.

The next step during the development of a social robot is deciding how these multimodal expressions will be generated. Although different approaches can be followed, in this dissertation two main groups of systems have been considered: those that rely on predefined expressions [51, 111], or those that learn how to generate expressions from scratch [112, 113]. While handcrafting all the expressions ensures that they suit perfectly the situation they have been designed for, it also constraints the possible responses that the system can execute. On the other hand, although the gestures obtained through automatic generation might not have the quality of handcrafted expressions, it makes it easier to extend the system to new domains, and also increases the variability of the robot's responses. Choosing between one approach or the other depends on the tasks that the robot in which the system is going to be integrated will have to perform. Manual design of expressions is a viable solution for robots that will be constrained to specific domains (for example, a shopkeeper robot), while an expressiveness system that automatically generates behaviours might be the most suited for general purposed robots. But in either case, individually selecting the appropriate actions (commands sent through one of the communication channels) that will be conveyed through each channel is not enough to guarantee a human-like expressiveness.

Multimodal expressions should not be understood as a collection of actions conveyed through different communicative channels that are performed at the same time, but as a single unit where these actions are connected to each other. This means that roboticists not only have to select the individual actions, but also design an appropriate strategy for combine them. Synchronization can be performed based exclusively on temporal information, which means that each action will have attached a start time, and that multimodal synchronization is achieved by properly adjusting these starting times. While this approach might work perfectly in virtual environments, in physical platforms is harder to implement, as different factors (for example, the inertia generated during

motions) might cause that the duration of a given action deviates from the estimated one. A possible solution for this problem is to connect actions from different modalities, so the beginning of an action is conditioned on the state of a different one. The most common example of this approach is to tie the start of gestures to specific points in the speech, so they are triggered when certain words are uttered. The advantage of this solution is that it accounts for any deviation from the planned execution of the expression.

4.1.2 Modulation of a robot's expressiveness

While the steps described above can lead to an expressiveness system that could be used to successfully complete the robot's tasks, functional behaviours (behaviours designed for achieving a specific goal) might not guarantee that the communication with a social robot is completely natural (as stated in previous chapters, a natural interaction is that in which the robot abides by the social rules of the context and meets the expectations generated on the human user). For example, although an automated telephone assistant for scheduling an appointment can perform all the necessary actions to complete its task successfully, it is not able to interact in a human-like manner, as users are able to recognize that there are missing elements of communication.

According to Buck and VanLear [114], communication can be decomposed into two different components. On one hand, symbolic communication has the goal of conveying a specific message to the other peer. This means that, during an interaction, the communicative goals of the participants in the interaction will be achieved thanks to this component. On the other hand, spontaneous communication is the act of non-intentionally conveying motivational or emotional states using a biologically shared signal system. This type of communication is non-propositional, this is, cannot be analysed as something that is true or false, as it is a reaction of an internal state. Messages conveyed by the peers in an interaction combine both aspects of communication. For example, if a person tells a friend *"I just bought a new car!"* while pointing at a car parked in the street, both the utterance and the gesture are examples of symbolic communication, as they are used to transmit a particular message. On the other hand, features like the pitch of the voice, or the speed of the motions, while not being essential for conveying this message, can indicate that the speaker is enthusiastic about this purchase, and thus can be used to infer his/her internal state. This indicates that robots should also be able to use both components during interactions with humans.

In robotic systems that rely on a library of handcrafted expressions, a possible solution for integrating symbolic and spontaneous communication could be to design multiple versions of

each expression that the robot can use, each version with variations that convey different internal states. But this solution might show scalability problems, as the number of expressions required would be increased a number of times equal to all the possible states of the robot. A second solution could be to design individual expressions for each internal state, and then merge the symbolic and spontaneous communication. Again, this approach has its disadvantages, as fusing expressions might not be a trivial task, and in any case this approach would display discrete states. A third approach can solve these issues: designing an strategy for an appropriate modulation of the robot's expressions. Therefore, developers would design the functional behaviours, while the spontaneous aspect of communication would be generated internally by the system, through the parametrization of different aspects of the expression (motion speed or amplitude, for example). This last approach has been supported by several researchers [115, 116].

Summarizing the information provided in this section, in order to design the expressiveness for a social robot (assuming that the embodiment has already been designed), several problems have to be overcome:

1. The definition of a model that describes the expressions of the robot. This model has to provide a description for multimodal actions, as well as a strategy for synchronizing these actions.

2. The development of a module that can schedule and execute the expressions of the robot on time.

3. The implementation of a strategy for conveying non-task related information, like the internal state of the robot.

While the first two steps are directed to endow the robot with the ability to engage in symbolic communication, the last one tries to extend these interaction abilities with the addition of spontaneous communication. In this chapter, a state machine-based model for multimodal expressions will be introduced, along with the Expression Manager, the module that will schedule and execute these expressions. The proposed model includes a series of methods for modifying the expressions of the robot in runtime, in order to adapt them to different internal states. As a proof of concept, these techniques have been used to convey different moods and emotions during interactions. Figure 4.1 highlights in red the part of the architecture introduced in Chapter 2 where the work that will be presented in this chapter has been integrated.

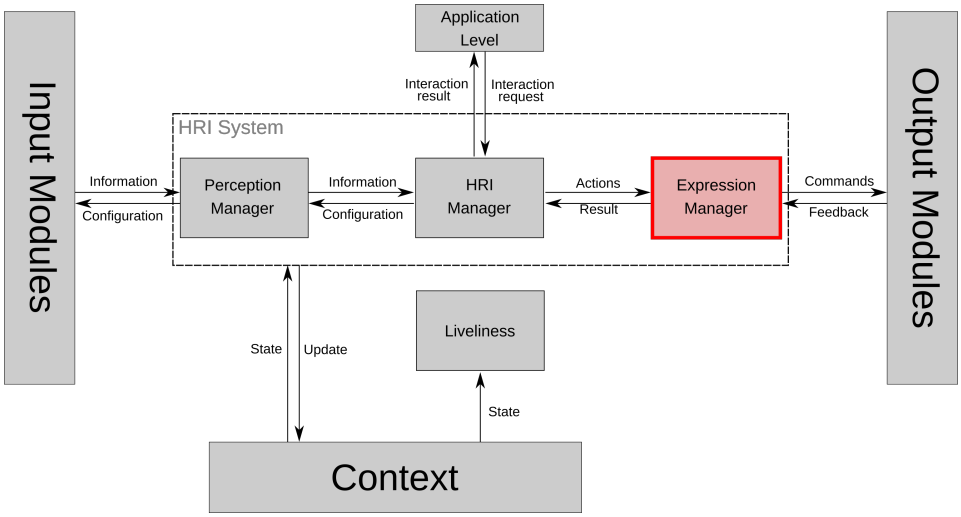


Figure 4.1: Diagram representing the software architecture described in Chapter 2. The work developed in this chapter of the dissertation focuses on the module of the architecture highlighted in red.

4.1.3 Objectives

The main goal defined in this chapter of the dissertation is to **endow a social robot with the ability to generate a human-like expressiveness**. This task has been divided into two main objectives:

1. **The development of a model for describing multimodal expressions to be performed by a social robot.** This model has to describe each of the actions conveyed through the different modes of interaction involved in the gesture, as well as the relationships between them.

2. **The development of an expressiveness management module that will schedule and execute the expressions that have been designed using the proposed model.** This module will be also in charge of managing any possible problem that might arise regarding the robot's expressiveness (multiple expressions being requested at the same time, or two or more gestures requiring the use of the same interface, for example).

Because these are goals with a high complexity, they have been decoupled in a series of subgoals that are easier to achieve, each of them related to a different facet of the global goals:

1. Expressions have to be easy to design, and creating them should not require specialized knowledge. The objective is that people with other backgrounds that might result helpful (for example, people with an artistic background) can participate in the design process without needing any sort of technical background.

2. The proposed system has to be able to modulate expressions, so they can be used to convey different internal states of the robot without having to design multiple versions of each gesture.

3. In line with the previous subgoal, users interacting with the robot should be able to distinguish the different internal states conveyed by the robot. An appropriate modulation strategy should lead to an improvement of the user's perception of the robot.

4. Expressions have to incorporate any combination of the interaction modes provided by the robotic platform. The expressiveness system will ensure that no conflicts arise regarding the use of the different output channels.

5. The system has to be able to manage the execution of multiple expressions at the same time. Appropriate strategies will be designed to manage incompatibilities between gestures. The idea is to combine expressions that are used for symbolic communication (conveying a specific message) with other types of gestures (for example, expressions oriented to increase the animacy of the robot).

6. Besides including basic actions (single commands sent to the output interfaces), the proposed system has to allow adding complex skills to the expressions. An example would be coupling the speech of the robot with a face tracking system. Expressions will control the activation and deactivation of these complex skills, which can be designed externally.

7. The performance of the system has to allow the robot to perform expressions in a time such that *feels natural* to the user interacting with the robot. This means that the maximum delay in the execution of expressions abides by the temporal constraints that affect human communication.

4.1.4 Overview of the chapter

This chapter of the dissertation will be divided in the following sections:

- Section 4.2: This section presents a review of works were models for expressiveness for communicative agents are described. This review will focus on systems that rely on a library of predefined expressions. Next, three key features are defined, and a comparison of all discussed works based on these features is conducted. Finally, this section presents the main similarities and differences between the proposed approach and the rest of works.

- Section 4.3: Here, the theoretical foundations of human expressiveness are presented, dividing it into verbal and non-verbal communication. For the non-verbal aspects of human expressiveness, this section focuses on those communication modes that can be used in our robots. The last

part of the section discuses the basis of affect state expression in humans, focussing on the different communicative interfaces that the robots' used in this dissertation are endowed with.

- Section 4.4: In this section, the expressiveness model developed in this thesis is first presented from a theoretical point of view. The key elements and features of the expressiveness model will also be introduced.

- Section 4.5: Here, this manuscript introduces the Expression Manager, the expressiveness management module that has been developed. An overview of all the features that it provides is presented from a technical point of view, along with the implementation of the model that describes the robot's expressions, and the tools used to create new gestures.

- Section 4.6: This section introduces the experiments conducted in order to evaluate the proposed expressiveness architecture. First, the experimental setups are presented, followed for a description of the tools used in the study (in this case, the description of the questionnaires that participants had to complete during the experiment), and a presentation of the demographic data of the participants (age, gender, education, and familiarity with technology/robotics). The experimental results are analysed and the overall performance of the proposed approach is discussed.

- Section 4.7: This last section contains the conclusions extracted from the development of an expression model for social robots. The goals that are presented in this introduction are evaluated in order to assess if they have been achieved, and the contributions of the work that has been described in this chapter are highlighted.

4.2 State of the Art

In this section, a review of relevant works in the field of expressiveness design is conducted, in order to highlight the similarities and differences between the solution presented in this dissertation and the current state of the art. This review focuses on works that are applied in the field of robotics, although it also includes a few researches that design expressiveness for virtual characters and environments, as in most cases, those systems could be adapted to be integrated in a real robotic platform.

As a general taxonomy for how to classify expressiveness design approaches, this dissertation considers that the main differential factor among works is how the gestures that the agent can perform are generated. In general, two categories were considered: (i) approaches that rely on a handcrafted library of expressions, and (ii) approaches that generate the different modalities for

the expression on runtime. The former have the advantage of allowing the developers to tailor the expressions of the robot so they convey the exact message and internal state that the situation requires. On the other hand, they can lack variability, as the possible actions of the robot are limited to a finite number of expressions. Automatic generation of expressions allows for a bigger variability, although the generated gestures might be perceived as being more generic.

In this dissertation, the expressiveness architecture developed follows the first paradigm presented. The reason behind this decision is that it gives freedom to application developers to define the robot's expressions in as much detail as required. Because of this, the analysis of the state of the art performed in this section will focus on expressiveness architectures that rely on libraries of predefined actions. While works in this section use different terms to refer to the communicative actions performed by the robot (e.g. expressions, behaviours...), in this analysis the terms expression or gesture will be used in general, for the sake of consistency.

In 2006, Kopp et al. [117] presented the SAIBA model, a multimodal behaviour generation framework for Embodied Conversational Agents. The goal was to develop a common specification for multimodal generation skills that would allow researchers to combine works that involve different aspects of multimodal behaviours. As part of the SAIBA framework, two representation languages were proposed: the Behavior Markup Language, or BML and the Function Markup Language, or FML, both application and domain-independent.

The proposed framework divides the generation of multimodal output in three stages: (i) intent planning, (ii) behaviour planning, and (iii) behaviour realization. The stages are sequential, and each provides feedback to the earlier stage. The framework focuses on defining the communication between stages, while the processing performed in each stage is treated as a *black box*, and left to each developer. This allows for a modular architecture where solutions proposed by different researchers for each stage can be combined without the need for complex modifications. The interface between the first two stages, specified with FML, describes the communicative and expressive intention of the system, while the interface between the second and third stages defines the multimodal behaviour that the last stage has to realize, and is specified in BML. Although the realization of this behaviour is going to depend on the particularities of the output modules of the system (for example, the realization engine can generate movements from scratch, or rely on a library of predefined animations), the BML is independent from the actual implementation of the behaviour realization stage, and instead provide a general description of the behaviour, from the actions involved and their temporal synchronisation to a detailed description of said behaviour's form. In this language, each

behaviour has an unique identifier that can be used to reference it. This can be used to synchronize two behaviours by specifying an identifier and a synchronization point, i.e. the points of alignment between them.

Along the years, developers have proposed expressiveness approaches that follow the SAIBA model. Anh et al. [118] presented a model for generating communicative gestures paired with speech for humanoid robots. This model was developed as an extension of the GRETA framework [119], a platform for virtual agents that follows the SAIBA framework [117]. It is divided in three modules: the intent planner, the behaviour planner, and the behaviour realizer. In this work, the authors have developed a new behaviour realizer for the GRETA system.

The system presented in this work was developed to control both virtual agents and physical robots. For this, two different lexicons were developed, one for each type of platform. They contain a library of symbolic gestures that describe the stroke phase for each gesture (the part of the gesture conveying meaning), represented as a sequence of key poses. The rest of the animation is then generated automatically by the system. A year later, van Welbergen et al. [120] presented the AsapRealizer, a behaviour realizer for achieving fluent incremental behaviour generation. It combines incremental multimodal behaviour generation with interactional coordination. This realizer allows for dynamically adapting behaviours already in execution without modifying the original specification constraints. The AsapRealizer module acts as the behaviour realizer under the SAIBA framework [117].

An execution engine runs the generated plan, while continuously apply modifications to the shape or the timing of the actions in the plan representation. These changes can be caused by events predicted by a set of anticipators and exert top-down plan or behaviour modifications. In this system, the generation of BML blocks and their individual behaviours is managed by two state machines: a behaviour state machine and a BML block state machine. Behaviours can be interrupted at any time. Also, in order to co-articulate behaviours, each of them has a priority that will be updated depending on their state. The engine uses this priority to decide which behaviour should take control over a given output interface. In 2016, Ribeiro et al. [121] presented the SERA (Socially Expressive Robotics Architecture) ecosystem, a model developed for combining an artificial intelligence agent with a robotic embodiment in Human-Robot Interaction tasks.The SERA ecosystem is divided in three levels (following the SAIBA framework [117]): the Decision Making module, the Behaviour Manager, and a series of output modules that represent the behaviour realization level in SAIBA. These include a text-to-speech engine, an animation engine called Nutty Tracks, and a multimedia

application. In this approach, multimodal actions are stored in XML files, containing speech information and markups for multimodal actions. Ribeiro et al. extend the SAIBA framework with the addition of the user's perception.

Animation programs in Nutty Tracks follow a box-flow type of interface, in which a set of animation controllers are connected sequentially to compose animations, either procedurally or based on a set of predefined animations. This sequence is then separated into a hierarchy of layers that can be activated or deactivated in runtime. The animation controllers can be programmed separately and then added to the animation engine as plug-ins. The body structure of the robot and the configuration and constraints of the different joints are stored in a robot-specific plug-in, which will be used to execute the animation frames in the robot.

While there is a fair amount of developers that decided to follow the SAIBA model for developing their expressiveness systems, other authors decided to follow other paths. For example, Meena et al. [122] proposed in 2012 a method for integrating speech and gestures (hand and head movements, and gaze following) to enhance interactions between humans and robots. The interaction system is based on WikiTalk, a speech-based dialogue system that can use Wikipedia as a knowledge database. The non-verbal gestures designed are aimed at marking the end of sentences and paragraphs, highlighting potential new conversation topics, or backchannelling.

In the version presented in [122], the system synchronizes speech and gestures by computing the average number of words that should be uttered before triggering the gesture, so the key pose coincides with the content word. The duration parameter of a gesture is then obtained based on the duration of the gesture's template and the length of the utterance. Punctuation signs and the structural details of the Wikipedia article being discussed are used to time the turn-management gestures. Trajectories for the gestures are the result of the interpolation of a series of poses. A year later, Salem et al. [123] presented the first closed-loop approach for generating speech and gestures for a humanoid robot. Modality synchronization is achieved by a scheduler module based on two features: a experimentally fitted forward model and a feedback-based adaptation mechanism. The former predicts the time required for gesture preparation, while the latter serves as an adjustment mechanism for cross-modal adaptation. In this approach, multimodal utterances are specified using an XML-based markup language, where only the stroke of the gestures is described in these utterances.

In 2015, Alonso et al. [51] proposed a multimodal fission module that receives a series of actions selected by the robot's dialogue manager and relies them to the output interfaces. This part of the

dialogue system is also in charge of the temporal synchronization of all the actions. The actions sent to the fission mode can be unimodal (an utterance, for example), or a multimodal expression. These are represented as an ordered sequence of unimodal actions stored in a XML file. Each action indicates the output interface, the configuration of the action (for example, the text that has to be sent to the TTS, or the final position of a joint), and a starting point, relative to the start of the expression. More recently, in 2017, Hemminahaus et al. [124] proposed a method for generating multimodal behaviours for assistive robots. Reinforcement learning is used to map high-level behaviours to low-level actions that have to be executed by the robot. The proposed system selects the optimal high-level behaviour based on the state of the interaction and the user, and then selects the appropriate actions for completing this behaviour. The work of the authors in [124] focuses on studying if the proposed method is able to guide the attention of the user towards a specific object in the environment.

Once the current affective and attentive state of the user has been determined, the system selects the more appropriate high-level function for accomplishing the intended communicative goal. This behaviour is then mapped into a series of primitives that, in turn, select and synchronize the actions to perform. They can be arranged into nested or sequential structures. The control layer of the robot executes the generated structures. Instead of generating robot-specific commands, the system can also describe the required instructions using BML, which will then be interpreted and executed by a behaviour realization tool. Q-Learning is used to improve the selection of these low level actions, to adapt the expressiveness of the robot to its experiences in the interaction. A year later, Ravenet et al. [125] presented a model for automatic production of communicative gestures for embodied virtual agents. The proposed approach is based on the concept of *Image Schemes*, recurring reasoning patterns used to map conceptual representations that ground abstract and concrete concepts between different domains. These image schemes will be used as a common ground between the verbal and non-verbal modalities. The system receives the speech that the agent has to utter, infers the image schemes from the surface text of the utterance, and then generates an appropriate set of gestures.

The first element in the architecture is the Image Schema extractor, a module that analyses the speech that has been generated for the agent, identifies the underlying schemes and aligns them with the utterance. The individual words in the utterance go through a disambiguation process first, and then each word is assigned to a synonym set (set of words with the same meaning). Then, the hypernymic relations of the set are used to find a synonym for each word that is connected to one of the image schemes proposed by the authors. Next, the aligned set of schemes is sent to the gesture modeller. For each schema, the module selects the proper gesture invariant (features used for conveying meaning) and builds the gesture as a combination of a beginning and end

phases parametrized to reflect this invariant. Finally, once the set of gestures have been defined and parametrized, the last module in the architecture combines them with either a hold or an intermediate pose to produce the final animation. One of the gestures is selected to be the main one, and the others are modified so they present the same features, except for those that have to remain invariant. Also on 2018, Balit et al. [126] presented the PEAR framework for prototyping expressive animated robots. This framework is based on Blender, an open-source 3D creation software. The system maps the state of an object animated using Blender to the actuators of a robot (in particular, the motors and a screen used to display facial expressions, icons, text, etc...). The architecture of the PEAR framework is divided in two main levels: (i) the dispatcher is in charge of the communications with Blender, and (ii) the controllers are the modules that communicate with the actuators.

The dispatcher is the main module in the proposed framework. It uses a mapping file to translate the modifications applied to virtual objects in Blender into a real world effect over the robot (for example, the position of a specific object could be translated into a specific motion of a motor). Developers can modify these virtual objects by hand, thus allowing for a teleoperation of the robot, or animate them using Blender's animation capabilities. In 2020, Gomez et al. [111] presented their approach to how to design the expressiveness for the Haru tabletop robot. The expressiveness is designed using animation techniques and expertise extracted from film-making, and then is transferred to the robot. Namely, the proposed method uses the *12 Principles of Animation* [127]. Due to the morphology of the robot, the primary non-verbal communication interface is Haru's eyes, although the designed expressions also include body motions, sound, and LED patterns. Each mode of communication is animated separately. After the design phase, the multimedia elements (videos for the eyes, audio for the sound...), and the trajectory of the joints are exported from the design suite, packaged into a routine file, and transferred to the robot in a way that both complies with the required physical constraints and maintains the integrity of the principles and techniques of the design.

On top of proposing methods for managing the expressiveness for robotic platforms, some authors have also focused on how this expressiveness can be used to convey the internal state of the robot. Among the different states that could be conveyed, one of the most prominent is the affect state. In this line, Xu et al. [115] presented in 2013 a method for endowing a robot with the ability to express mood while performing functional behaviours (behaviours oriented to completing a task), through the modification of the gestures' appearance (both pose and motion). The proposed behaviour model follows a layered approach, where the affect is used to alter pose and motion parameters, and then these parameters change how a gesture selected by the system's task scheduler is performed. The appearance of these gestures is described in behaviour profiles. More recently, in

2017, Bertacchini et al. [128] presented a robotic shopping assistant using a NAO robot. The design of the system was focused on endowing the robot with the ability to analyse multimodal perception data about the user's emotional state and other related information (taste, cultural background...), and also the ability to express its own emotional state. In this work, the developers relied on NAO's software environment for developing all the expressiveness capabilities of the robot. In NAO, body motions are programmed using Choregraphe [129], a graphical interface for programming robot behaviours in the platforms developed by SoftBank Robotics. This application provides a series of both low and high-level actions (walk, basic postures, speech synthesis, LED control, etc...) that can be combined into complex behaviours. New actions can be programmed in python and then added to the library in order to add new capabilities to the robot. In the interface, the library of behaviours is represented by blocks that can be dragged and connected one to another. The new behaviours can then be executed on a virtual representation of the robot or on a real platform. Time control is implemented through a timeline that allows to specify the length of each action contained in the behaviour.

In 2018, Van de Perre et al. [130] proposed a gesture generation method for social robots. Two modes of execution were developed inside this method: (i) the *block mode* allows to create gestures that rely on a correct position of the arm, and (ii) the *end-effector* mode is used for gestures where the position of the end-effector becomes the critical factor (for example, pointing expressions). Both modes are combined in order to create deitic gestures and emotional expressions. Under this approach, functional behaviours can be modified so they convey different affect states.

Also in 2018, Churamani et al. [131] proposed a neural model for multimodal affect recognition, analysis and behaviour modelling. Once the mood of the robot has been correctly defined, it is conveyed to the user through facial expressions. In the robotic platform used in [131], a LED projection system located inside the robot's head is used to display the different expressions, which are defined by a set of wavelet parameters that represent the position of the eyebrows and mouth. The system uses a Deep Deterministic Policy Gradient based actor-critic architecture to learn the optimum policy for selecting the most appropriate wavelet parameters based on the emotional state of the robot. A pair formed by the state of the robot and the action generated is sent to a critic network that predicts the pair's Q-value based on the reward received for executing that action in that state. The reward function encourages symmetry in the facial expression. The authors also designed emotion-specific rules based on studies conducted with users about the expression of affect states in that particular robot. A year later, in 2019, Desai et al. [132] proposed a system for developing expressive behaviours by editing their control parameters in a semantic space. In the

version presented in [132], the system supports semantic design robot motions that convey one of six possible emotions: happy, sad, angry, scared, surprised and disgusted. This framework consists of four steps: (i) generate a dataset of expressive motions; (ii) use a crowd-powered framework for evaluating how well these motions express a particular emotion; (iii) find the relationships between motion parameters and the expression's emotional perception through the use of a regression analysis module that relies on the evaluations conducted in the previous step; (iv) finally, create new expressions using a design tool that relies on the relationships found in the previous step.

That same year, Mier et al. [133] proposed a method for generating expressive motions for social robots. Designers with a background in animation created a series of motion animations that displayed different expressions with various intensities. Then, these animations are used to generate new expressions through an interpolation process that combines them.

In this work, the expressions considered are *happiness*, *shyness*, *sadness*, *agreement*, and *disagreement*, and the following combinations are allowed: *happy-agree*, *shy-agree*, *shy-disagree*, and *sad-disagree*. Each expression has a parameter that indicates the intensity of the internal state conveyed (for example, how happy a particular expression looks), rated on a 1 to 10 scale, and only expressions with the same intensity value can be combined. Both the expression and the intensity are mapped to a set of coordinates in a 2D wheel of emotion, proposed by Plutchik [134]. In 2020, Suguitan et al. [116] proposed a neural network-based method for modifying affective robot motions. This approach tries to overcome the limitations of expressiveness approaches based on limited sets of gestures by adapting the expressions so they can express different affect states. In order to do this, the system uses variational autoencoders, classification networks and latent space editing methods. The approach learns a low-dimension latent representation of the affective motions, which is used to classify the movements by their intended emotion. Arousal and valence variations are performed over the latent representation of the motion, in order to modify the affect state conveyed. Finally, the new motion is reconstructed from the modified latent representation.

In order to map the low-dimension latent space into a 2D arousal-valence space, the system uses linear regression. The first step is to map the centroids of the emotions classes in the latent space. Next, the centroid is recalculated to reduce the effect of motion samples that belong to one class, but can be confused with a different class. Finally, the linear regression model transforms the movements into the arousal-valence space. To modify the expression, first the arousal-valence representations of the data samples are ranked on a high-low scale for valence and arousal features. Then, for each feature, an attribute vector is computed. These vectors are then used to change the arousal and

valence of the motion. In their work, the authors also presented a graphical interface that allows to visualize the arousal-valence representation of the motions. The different emotion classes are highlighted using coloured dots (green for happiness, red for anger, and blue for sadness). Developers can select a movement in the interface, and modify its valence and arousal using sliders, which will then translate into a variation of the feature weights that are used to modify the latent representation of the movement. Another possibility is to directly select the new emotion, which will lead the system to update the sliders. After the latent representation of the motion has been modified, the decoder reconstructs the gesture in the form of motion trajectories.

4.2.1 Comparison between approaches

All the works presented in this section of the manuscript were compared according to a series of characteristics that were considered relevant for Human-Robot Interaction. First, for interactions to feel natural to the human user, the robot should be able to blend multiple modes of interaction in order to accomplish a specific communicative goal, taking advantage of the strengths of different output sources. Endowing a social robot with this ability gives interactions a higher flexibility and can help to enhance the experience of the user. The addition of new communication sources that are not commonly used in day-to-day human-human interactions (patterns of coloured LED, or multimedia content displayed on screens, for example) give robots more tools to convey communicative goals or internal states in a way that is engaging for the user. Thus, having a expressiveness system that can work with multiple sources of information and coordinate all the modalities with precision becomes a highly desirable feature when developing a human-robot interaction architecture for social robots. In this context, multimodality is understood as the feature that indicates the different output channels that an approach can use to communicate a message, and how the actions for each of the interaction modes are synchronized among them.

Second, when creating a new expressiveness management strategy, one of the first decisions that has to be made is the format that will be used to define the expressions. This includes aspects like if an external GUI or software will be used for the design process, or the structure that the gestures are going to take, among others. The goal is to select a gesture format that is advanced enough to describe complex multimodal expressions, but can be built with the minimal amount of effort and skill required. This can help to bring into the design process people with expertise in areas like animation, without the need for being familiar with how the interaction system is built or even having a programming background. This idea will be referred to in this section as *gesture design*, and is defined as the way new expressions can be added to the robot's expressiveness. It analyses the

format used to represent expressions, and the possibility of using external design programs to create new gestures.

However, even if the robot can use a large set of multimodal expressions that have been expertly crafted, the interactions can still feel repetitive if the robot always performs the same actions in each situation. The expressiveness of a robot should be able to reflect different internal states and adapt to the context of the interaction, generating different expressions for conveying the same information in a variety of situations. One of the most common examples is showing multiple emotions with the same gesture. This feature is known in this thesis as *adaptability*, and describes how the expressions in the proposed approach can be adapted to different circumstances that arise in the dialogue. This does not focus on how a specific expression is modified to reflect a situation in the interaction, but on how the expressiveness system as a whole can adapt to these circumstances.

The following sections will present a comparison of all the works reviewed according to the features presented above, in order to identify the differences between them. A summary of the results of this comparison can be seen in Table 4.1.

4.2.1.1 Multimodality

After analysing all the works presented above, it can be observed that the mode of communication that almost all the approaches include is body motion, followed closely by speech (all but three approaches use both motions and speech). The exception is the work of Churamani et al. [131], which uses facial expressions through LED-based features (eyebrows an mouth). LED patterns, gaze, facial expressions, and screens to play multimedia content are less common in the works reviewed, as those modalities are more closely related to the design of the robotic platform. For example, the work of Suguitan et al. [116] and Desai et al. [132] focus exclusively on body motions, while authors like Balit et al. [126] and Gomez et al. [111] introduce the use of LED. The latter work separates itself from the rest of the works due to the fact that the eyes, both the gaze and the motion, represent the main communicative feature of their robotic platform. Using the same robotic platform, Mier et al. [133] decided to ignore the other modalities and focus exclusively on motions. Finally, the work of Alonso et al. [51] has to be highlighted here, as it was developed combining speech, LED patterns, motions, gaze (through animations displayed on two TFT screens) and the use of a touch screen for both displaying multimedia content and also receiving inputs from the user.

Reference	Gesture design	Adaptability	Multimodality
[117]	Gestures specified in BML format. Single actions or combined	Not mentioned	Voice, body motion, facial expression. Tags used to indicate synchronization
[118]	Stroke poses stored in a lexicon, gesture built through interpolation	Not mentioned	Speech, gestures
[120]	Multimodal behaviour described in BML	Shape and timing of expressions adapted before and during execution	Voice, body motion, facial expression. Tags used to indicate synchronization
[122]	Poses interpolated with b-spline algorithm	Not mentioned	Voice, gestures
[123]	Stroke described in the utterance, motions to and from that pose generated	Speech-gesture synchronization corrected based on gesture's timing	Voice, gestures
[115]	Gestures created as behaviour functions, non-task features adapted to the robot's state	Behaviors parametrized through motion and pose parameters	Voice, gestures
[51]	Expressions stored in xml files as a list of actions. Actions can be requested individually	Not mentioned	Gaze, Voice, Motions, LED, multimedia content
[121]	Speech and multimodal action markups stored in Skene Utterances. Animations designed externally or generated procedurally	Not mentioned (animations can be moduled)	Gaze, Speech, body motions, animations
[128]	Gestures designed through Choregraph and then combined with other modalities	Gestures adapted based on the user's emotion	Voice, gestures, LED

Table 4.1: Comparison among the works presented in this section. Each approach has been evaluated according to gesture design, adaptability, and multi-modality.

Reference	Gesture design	Adaptability	Multimodality
[124]	Reinforcement learning for action selection, based on interaction goal	Reinforcement learning for adapting action selection	Voice, gaze, gestures, facial expression
[125]	Schemas into invariants into gestures	Not mentioned	Voice, motions
[130]	Target gestures predefined (block mode). final position of the chain defined, pose generated (end-effector mode)	Speed and amplitude used to generate affective behaviours	Gestures
[131]	Facial features generated based on affect	Not mentioned	Facial expression
[126]	Designed in blender, played in the robot	Not mentioned	Not mentioned
[132]	Gestures designed through a interface and adapted based on the desired emotion	Gestures desinged to convey affect, not adaptables	Motions
[133]	Animations handcrafted for certain states, combined through interpolation	Not mentioned	Motions
[111]	Modalities designed independently externally, packaged, and transferred to the robot	Not mentioned	Body motions, LED, voice, eyes
[116]	Head motions designed through a phone app that copies the movement of the phone	Motions can be altered to convey different affect states	Body motions

Table 4.1: Comparison among the works presented in this section. Each approach has been evaluated according to gesture design, adaptability, and multi-modality.

Finally, regarding the synchronization of the different output modalities, the approaches followed usually depend on the strategy used to design the expression. Works where the expressions are handcrafted tend to rely on assigning time points to the different actions. The work of Alonso et al.[51] is an example of this. Other approaches use the speech as the central element, and synchronize the other modalities to points or words in the utterance. This can be seen in the work of Ribeiro et al. [121], Anh et al. [118], or van Welbergen et al. [120], among others. Finally, approaches that use external animation design software to create all the modalities of the expression can rely also on the features of the animation software to combine the different actions. This can be seen in the work of Gomez et al. [111].

4.2.1.2 Gesture design

This feature was the one that showed the highest variability among the reviewed works. Kopp et al. [117] has to be highlighted here, as the authors proposed a structure for expressiveness management systems and a format for describing expressions that became very popular in the field. For example, the approaches proposed by Ribeiro et al. [121], Anh et al. [118], or van Welbergen et al. [120] are compliant with the SAIBA framework defined in Kopp's work. Overall, a big diversity on how gestures are designed can be observed among the works presented in this section. When designing body motions, specially if the morphology of the robot is not simple (for example, humanoid robots with a high number of degrees of freedom), using animation software is a highly used solution. Examples of this can be seen in the works of Balit et al. [126], Ribeiro et al. [121], or Gomez et al. [111]. The work presented by Desai et al. [132] also relies on a interface for desinging the expressions, but in this case is custom made, specifically designed to suit their needs. Also, a very popular platform in the field of robotics is the Nao robot, developed by SoftBank robotics, which provides a proprietary interface to generate the behaviours of the system, the Choregraphe framework. The use of this software can be seen in the work of Bertacchini et al. [128]. Finally, the work of Suguitan et al. [116] proposes an unique solution for the design of motions: use a phone app to capture the motion of the device and then transfer it to the robot.

If the comparison between approaches is focused on how the expression is structured, an approach that can be found in multiple works is the use of markup languages to define a script that defines when each action has to be performed. For example, in the work of Ribeiro et al. [121], they use a xml-based structure that contains the speech of the robot and mark-ups that indicate where the other modalities have to be executed. A similar approach was followed in the work of Salem et al. [123], which defines multimodal utterances that indicate the point of the speech that has to be used

to align the stroke phase of the gesture, or the work of Alonso et al. [51], which created multimodal expressions using XML files that store a sequence of actions labelled with the time point in which they have to be performed. Other approaches, like the ones presented by Ravenet et al. [125], Meena et al. [122], and Anh et al. [118] maintain a library of poses, while the actual motions are synthesized in runtime as an interpolation of the poses. Ravenet's work defines a series of gesture invariables to constraint this synthesis process. The approach presented by Van de Perre et al. [130] also defined a library of poses in one of the two proposed modes of execution, while in the other the trajectory for all joints is generated based on the position of the end-effector. The system presented by Mier et al. [133] introduces a library of animations that represent different internal states of the robot. New animations that convey multiple states at once can be generated as an interpolation of two examples from this dataset. Finally, three works propose a more unique approach to design their expressions. Xu et al. [115] handcrafted the expressions in behaviour profiles as behaviour functions, which are task-specific, and non-task behaviour parameters. Lastly, the work presented by Hemminahaus et al. [124] introduced the use of reinforcement learning to convert a communicative goal into a sequence of low-level multimodal actions.

4.2.1.3 Adaptability

The approaches presented by van Welbergen et al. [120] and Salem et al. [123] propose techniques for adapting the non-verbal side of the expression to circumstances that hinder the predefined synchronization of the multimodal behaviour. While the former proposes a solution where behaviours are generated incrementally, which allows the system to adapt the timing and shape of the gesture to changing circumstances in the interaction, the latter focuses only on the timing, and can introduce pauses on the speech to correct the synchronization between speech and motion if the gesture's execution deviates from the simulation computed beforehand (if the motion trajectories take longer or shorter to complete the preparatory stage of the gesture). Here is also interesting to mention the work of Hemminahaus et al. [124], which can adapt which actions are performed by the robot to achieve its goal using reinforcement learning. In this case, while the actions themselves are not modified, the combination of actions is improved during the interaction.

As stated before, one of the more common applications of gesture modulation techniques is the display of affect states. An example of this can be seen in the work of Xu et al. [115], where the task-specific aspects of the expressions are defined in behaviour profiles, while a set of behaviour parameters can be used to modulate the expressions without affecting the overall meaning of the gesture. This approach can be used for conveying different states of the robot (in their work, the

authors tested it for mood expression). A similar approach was proposed by Van de Perre et al. [130], where the body motions of the robot can be modified according to two dimensions: speed and amplitude. Similar to Xu et al., these two parameters are used to display affect states. Authors like Desai et al. [132] and Suguitan et al. [116] also proposed methods for generating emotional expressions. While Desai's approach allows the developer to define which emotion should a motion convey, and also use a library of motions to start the design process from, the work of Suguitan et al. proposes a method for automatically create versions of predefined motions with a change on the emotion being conveyed.

4.2.2 Comparison with the solution proposed in this thesis

The works of Alonso et al. [51], Ribeiro et al. [121], and Kopp et al. [117] use a similar approach for describing the behaviours that the robot will have to perform. They use different markup languages to represent the multimodal actions that will be performed, along with any information required to synchronize them. The advantage of this approach is that represents the expressions in a language that is easily understandable by humans, and thus it is easy to craft new expressions. But this type of solutions can complicate the process of synchronizing the different modalities, as expressions usually rely on temporal information for synchronization, which is susceptible of being affected by different factors, like the inertia of the motions. Another possibility is to include synchronization points in these expressions, to specify how the different actions connect to each other, but this again requires crafting specific synchronization methods. This thesis proposes the use of a structure for representing expressions that is at the same time easy to craft and visualize, and that takes care of the synchronization of actions internally.

In the work presented in this manuscript, expressions are modelled as state machine-like structures, where the actions that the robot has to perform are represented as states of this machine. On top of this, there are also states used specifically for controlling the flow of the expression, and perform synchronization tasks. That way, the synchronization between actions is represented with the transitions between states. For example, if the arm of the robot has to be risen after a utterance has been spoken, each action can be modelled as an individual state and the transition between states specifies the temporal relation between actions. Although the work of van Welbergen et al. [120] also presented the use of state machines for expressiveness, their work uses this structure to control the process of executing expressions, while here the state machine represents the internal structure of each expression. Opposite to traditional finite state machines, the proposed structures can maintain multiple states active at the same time, so the robot can perform multiple actions

at the same time. This approach presents some advantages. On one hand, it allows to combine different synchronization approaches observed throughout the works reviewed and use either time points to indicate the beginning of each individual action, or set the execution of these actions based on the completion of one or more previous steps of the gesture. Particular states have been developed to control the start point of an action, and to force the expression to wait until an action has been completed before continuing with the next one. Also, the solution proposed in this thesis also gives developers the ability to include aspects of traditional control logic into the development of expressions. For example, a gesture can include loops to specify the repetition of either individual actions or entire sections of the expression, or use branching to define multiple paths that the execution of the gesture can follow depending on some external variable. All the control functions have been encapsulated in independent blocks that can be shared among expressions, thus eliminating the need to program them again for each new gesture.

In order to simplify the design of new expressions, the proposed approach also provides a graphical interface for the creation of gestures, similar to the solutions proposed by Gomez et al. [111], or Desai et al. [132]. In this thesis, the graphical interface was included as a tool to simplify the process of crafting the expression files, instead of being a tool for animating the robot in a visual and direct way. Developers still need to follow a trial and error process to adjust the expressions in the proposed approach. The interface selected allows developers to create expressions by dragging and connecting blocks, which simplifies the design process and allows persons without a programming background to create or modify expressions without much effort. Also, the interface allows them to visualize the pipeline of the expression, making it easier to appreciate the connections between the different modalities used.

Regarding adaptability, works like the ones presented by Xu et al. [115] or Van de Perre et al. [130] propose to use external parameters to modify the appearance of the expressions developed in order to convey the different internal states of the robot. While Xu et al. defined multiple behaviour parameters, Van de Perre et al. focused on the modulation of the expression's speed and amplitude. While both approaches have their advantages, they also present weak points. Thus, having the possibility of combining multiple techniques can help to adapt the system to a larger amount of situations. Also, these techniques only modify the appearance of an expression, but cannot change the value of the actions contained, to increase its variability. This thesis proposes the combination of different modulation methods for altering features tied to both the symbolic and spontaneous aspects of the expression (for example, altering the pitch of the voice, or changing the sentence that the robot has to utter). One of the techniques included

follows the idea of Van de Perre et al. and modulates a multimodal expression on the speed and amplitude dimensions, while the second relies on modulation profiles to specify a more detailed adaptation of each interface. Also, a third technique was introduced to modify the configuration of any action in an expression on runtime. This not only allows to convey the internal state of the robot through an appropriate modulation strategy, but also allows the system to modify any action conducted by the robot in order to better adapt the expressions to the context of the interaction.

4.3 The anthropological foundations of expressiveness

According to some authors [74], communication is one of the key factors that define our society, and separates humans from the rest of the animals. Although other species are able to develop basic gestural communication systems, the thing that is completely unique to the human kind is the development of language. This communication system allows humans to describe any abstract mental concept in a way that can be understood by any other language knowledge, without the need of establishing first a common ground. On top of this, humans are able to use non-verbal behaviour (for example, body gestures or facial expressions), either on its own, or as a way to enhance the information conveyed by the speech. Due to the fact that social robots are designed to perform tasks that involve a large amount of interaction with humans, endowing them with the expressiveness capabilities that allow them to be perceived as social agents is an important problem in the area of Human-Robot Interaction. This section presents an analysis of both verbal and non-verbal communication in humans, and their ties to the task of robot expressiveness design. This analysis seeks to highlight a series of features involved in human multimodal communication that will be integrated in the expressiveness management approach proposed in this chapter.

4.3.1 Verbal communication

Although is has been largely discussed if other animals are able to produce intentional vocal displays that have a communicative purpose, something that cannot be argued about is the fact that humans are the only animals that have developed a vocal communication system as complex as language. According to the Cambridge Dictionary, language can be defined as *"a system of communication consisting of sound, words, and grammar"*. Different expressions of language can be distinguished, including writing (using a set of symbols to visually represent language) and speech (the vocal expression of language), among others. Language is learnt innately by children in early stages of

their development [135]. Around 6 months old, their gurgle starts to become babbling. A year later, they are able to utter around 50 words, and are able to understand three times that amount. By the time they are 2 years old, the characteristics of the language used in their environment (tone, accent, rhythm) can already be appreciated in their speech. By the time they turn 3 years old, children are usually able to utter complete sentences, and have a vocabulary of around 1000 words. This amount is increased tens of times during a person's lifespan. The ability for learning new languages fades with age, being easier for children than adults. Along the process of learning language, different challenges appear. For example, recognizing the borders of words in speech is an arduous task. According to the research of Saffran et al. [136], children overcome this problem by learning a probability-based combination of sounds, where they are able to learn that if a combination of sounds has a low probability, it might indicate the presence of a border between two words.

Speech production in humans is a complex process that involves the coordination of sets of muscles that control the lungs, larynx, and mouth [135]. The larynx plays a central role in this process. In order to produce sound, the air kept in the lungs is exhaled and goes through the larynx. In here, two bands of muscle, called vocal chords, vibrate due to the air that is passing through the larynx. On its own, this is not enough to generate sound. But if the vocal cords are tightened, then the vibrations will translate into sound. This is the reason why no sound is emitted while breathing. The pitch of the generated sound can be regulated through a variation of the tension applied over the vocal cords. This sound is then further modulated in the throat and mouth. In the last stage of the process, the components in the mouth transform sound into phonemes, the basic sounds used to build speech, through a series of fast shape changes. Although one could think that speech is exclusively tied to the auditory modality, in reality it also involves the visual channel. This is due to the fact that part of the articulatory organs used for generating speech are visible by the listener (i.e. the mouth). An example of this relationship is the *McGurk effect* [137]. It refers to the fact that speech recognition is affected by the visual modality as well. For example, if one phoneme can be identified through the auditory modality, and a second phoneme can be identified visually by focusing on the speaker's mouth motion, then the listener might not perceive neither, and instead perceive something new. Visual cues can also be used to focus attention on the speaker, and to help the listener to discriminate background noise from the speaker's voice [138].

4.3.2 Non-verbal communication

Non-verbal communication can be understood as the process of sending and receiving communicative messages that do not involve the use of words. This definition also includes the non-verbal aspects

of language, which are known as paralanguage. This discipline of communication was initially proposed by Charles Darwin [139]. In [140], Burgoon and Bacue presented that, according to some estimations [141, 142], around 60% of the information conveyed during an interaction is transmitted through non-verbal methods. There are five main abilities of non-verbal communication when paired with verbal messages [105]: (i) repetition, which reinforces the verbal message; (ii) contradicting the message conveyed by the speech (iii) substituting words in the verbal message; (iv) complementing what is being conveyed through the speech; and (v) accenting specific parts of the speech. While encoding verbal communication is a discrete process, as it is constrained by the words used, non-verbal signals can be encoded in a gradual or continuous way by changing the intensity of the message in order to convey different reference magnitudes [143]. For example if a non-verbal expression signalling stop is done at a high speed, it indicates the need for a more abrupt stop. In any case, although there is the possibility of adapting non-verbal behaviours so they adhere to different cultural or social conventions, or to achieve specific communicative goals, non-verbal communication tends to be a rather automatic process that relies on implicit knowledge that is innate to humans [144].

Non-verbal cues can be displayed through a large variety of communicative channels, and can be performed in ways that are more explicit or implicit. Types of non-verbal cues include body motion, which in turn can be decomposed into the movements performed with different parts of the body (hand gestures vs whole body expressions, for example), and also posture, which is static; the way in which participants in a conversation engage in eye contact; the display of certain facial expressions; the use of distance between speakers; non-verbal sounds; the use of touch in specific ways; and even the use of time itself (a discipline known as chronemics, which studies how people uses time to influence communication). But non-verbal messages can be conveyed as well in a less intentional manner. An example could be how wearing specific clothing can convey a certain social status or role that is going to have an effect on interactions between individuals (for example, wearing a lab coat and scrubs in a hospital might indicate that a person is a doctor or nurse, thus prompting different responses of people with whom they interact). Among all the possible types of non-verbal cues, this section will focus on the following: body motion and posture (kinesics), facial expressions, and paralanguage and prosody. The reason behind this is that the expressiveness capabilities of a robotic platform are constrained by the hardware it includes, and these modes of communication are the ones that the platforms used in this thesis share with humans.

4.3.2.1 Kinesics

According to the work of Birdwhistell [141], who is considered to be the founder of this research area [145], Kinesics can be defined as the study of human communication involving body motions and gestures. This not only includes actions where the body is in motion, but also static postures. There is no direct translation from body movements to verbal communication [145], as the meaning of the majority of body motions will show a high variance within and across cultures, while words have a fixed meaning that is known for any speaker of that particular language. Although motions can involve a large number of joints and body elements, when focusing on the use of motion in social interactions, the attention is usually focused on the terminations of the limbs. Specifically, most of the movement that is related to interaction is performed with the hands and the head, and thus research on body movement has paid special attention to these limbs.

In kinesics research, strategies for coding body motions have focused on *action* behaviours [145], which are discrete motions that do not have the purpose of positioning the body, do not require to be intentional, and have a distinguishable start and end, These actions are supported by *position* behaviours, which involve the static positioning of the body to convey a message. Body position tends to display a smaller variation between individuals, and their change is not very frequent. Usually, the position of each limb is not considered individually, but as a whole unit.

Gestures can be interpreted differently depending on their timing with regards to the situation in which they are performed, and context plays a big role in giving them a distinctive meaning. Also, on top of the configurations of the limbs and the shape and direction of movements, features like speed, acceleration, strength, etc... also can change how a specific gesture is perceived.

4.3.2.2 Facial expressions

A facial expression involves the positioning of the muscles of the face to convey a specific message. Even though elements of the face (for example the eyes) can have diverse communicative functions (examples would include establishing eye contact, or performing conventionalized facial expressions in the context of a sign language), seems that facial expressions are mainly associated with the display of emotional states. Voluntary facial expressions follow a series of socially learned norms that define how facial expressions should be regulated in social interactions, while purely emotional expressions are usually not performed on command [146].

Regarding the expression of affect states through facial expressions, there has been a long discussion about the nature of these expressions. On one side of the argument, the opinion is that facial expressions of emotion are universal, while the other side argues that they are cultural-specific. Ekman conducted a review of the evidences and counterarguments for both postures in [147]. The conclusions from Ekman's work state that it exists a universal connection between certain facial configurations and emotions being conveyed. The evidence does not indicate how many universal expressions exist for each emotion, or if all emotions have an associated universal expression. Cultural and individual differences exist and play a role on how emotions are expressed, both in the encoding and decoding aspects, and also on the social rules that govern the display of emotions.

4.3.2.3 Paralanguage and Prosody

Paralanguage, term coined by Trager in 1958 [148], is the component of non-verbal communication that rely on the modulation of the non-phonetic features of the voice to affect the message of the voice, or convey an affect state. The field that studies paralanguage is known as paralinguistics. Although some definitions of the term paralanguage consider also visual communication, this section will follow the criteria shown in the work of Schuller et al. [149] and include exclusively those aspects of paralanguage related to the modulation of the speech. These aspects are entangled in the word chain, and can denote the internal state of the speaker, his/her affect state, characteristics tied to the individual, and the like [149]. Schuller et al. [149] presented a more detailed list of traits that play a role in speech modulation, divided into three classes: (i) long term traits, which include biological aspects (age, body type, gender...), cultural traits, personality of the speaker, and other speaker-related idiosyncrasies; (ii) medium-term traits and states, which include temporary internal states like health or mood, and structural signals related to social, behavioural, and interactional domains; (iii) and short-term states, which include the speaking style and voice quality, emotional states (which are short-lived, opposite to moods) and other related states, like stress or politeness.

While some authors proposed that all non-verbal aspects of speech should be considered as paralanguage, others establish a distinction between paralinguistic and prosodic features [150]. Prosody can be defined as *the rhythmic and intonational aspect of language*, or as a set of variables that are used to differentiate vocal patterns. Prosodic features of the voice include tempo and rhythm, intonation, emphasis, and also the pauses made on the speech. Intonation refers to the variation of the pitch during spoken utterances.

4.3.3 Applying human communication features to a robot

From the review of the human communication abilities conducted above, several conclusions can be extracted and used to plan the integration of an expressiveness architecture in a robot. First of all, because speech is the only method of communication that is exclusive to humans, robots designed to interact with people should be endowed with the ability to speak as well. This might not only involve the integration of a speech generation module that creates the auditory component, but also the use of other visual cues to reinforce the information contained in the speech. This can involve techniques ranging from using visual cues that indicate that the robot is speaking (for example, a LED that is turned on while speech is being uttered) to modifying the shape of the mouth so it matches the words being uttered.

A second major conclusion is the importance of non-verbal displays during communication [141, 142]. Non-verbal communication should not be considered as a discrete mode of interaction (having predefined actions that are performed always the same way), but as a continuous system with an intensity that can be modified during an interaction. Because the situations under which non-verbal behaviours are used can be considered culture-specific, they should be adapted to the context in which the robot will be used. Also, the fact that there is an unintentional component to non-verbal communication that will have an effect on interactions and that is related to the appearance of the speaker (for example, the use of specific clothing to indicate a particular social role) could suggest the importance of adapting the appearance of the robot to the social role it will be taking.

Regarding the specific use of the different non-verbal modalities reviewed, there are also a series of conclusions that can be extracted:

1. The head, arms, and hands tend to be involved in the majority of motions used for interaction purposes. This suggests that the expressiveness architecture should prioritize the control of these limbs over other body motions (for example, gait).

2. In the study of kinesics, researchers tend to focus on groups of limbs, instead of individual motions. This could point towards the importance of combining multiple limbs in a single expression, instead of developing individual expressions for each limb.

3. A correct timing of the motions and a proper modulation of the motion parameters can help to modify how an expression is perceived, and adapt them to the particularities of a given interaction.

4. Facial expressions, and in particular the eyes, are commonly tied to the expression of affect states. This is a complex task, as each emotion can be tied to different features of the face. Also, while some emotional expressions can be considered universal, the cultural environment in which the robot will be integrated should be considered when developing these expressions.

5. During speech generation, not only the verbal content should be considered, but also the non-phonetic features of the voice. Based on the division proposed by Schuller et al. [149], the modulation of the voice could be used to display long term traits that can endow the robot with a personality, while modulation for conveying medium and short term traits could be used to adapt the voice to the context of the interaction.

6. As stated in Section 4.1.2, humans not only communicate using goal-oriented expressions, but they also transmit involuntarily their own internal state. This indicates the importance of considering also how to endow the robot with the ability to convey these states. While spontaneous communication involves an extensive range of states, this dissertation will focus on one of the most prevalent: affect states. Due to this, Section 4.3.4 will analyse how humans are able to express these states.

4.3.4 Expression of affect states in humans

How humans express emotions and moods is one of the earliest research questions in the field of human expressiveness. The origins of these studies can be traced back to the work of Darwin in his book *"The Expression of the Emotions in Man and Animals"* [139]. He considered that the communicative functions performed by emotional expressions presented an added adaptive value, on top of the biological advantages of emotion (preparing the body to react to a specific situation). In [151] Hess and Thibault discussed the principles presented by Darwin, and then presented some lines of research that found their root in Darwin's work. One of the research topics discussed has to do with the meaning of expressions. While Darwin considered that expressions were simple manifestations of emotional states, other researchers saw emotional expressions as cultural signs, based on the idea that facial expressions are culturally learned. Later theories proposed that emotion expressions are connected to behavioural intentions, and that they should be seen as communicative signals modulated by social norms. Another issue discussed in [151] is the existence of prototypical emotional expressions. While an argument in favour of this is the fact that the emotional expressions that Darwin, Ekman, and Friesen described tend to be recognized correctly, there is not the same amount of support for the idea that these expressions are actually displayed by people under that particular emotional state (there are works that report the existence of that connection for certain emotional states, while other works do not find any relation). A third research topic analysed is

the universality of emotions. While Darwin was convinced that emotions are universal, and this conclusion was supported by Ekman, several issues were raised about how the experiments were designed in the works defending universality. Other researchers argued that emotional displays are culture-related, based on the evidence pointing that people tend to recognize more accurately emotional expressions performed by individuals in their own cultural group.

Next, the expression of affect states will be decoupled according to the different communicative channels described in this section (speech, body motion and posture, and facial expressions). Regarding the use of facial expressions to display emotion, authors like Darwin [139] and Ekman [152] presented a description of those emotional expressions related to the emotions considered to be universal. In another work [153], Ekman assured that the measurement of emotional expressions should not be done focusing on specific facial features, but on their combination. The example he provided was that pulling down the lips' corners conveys sadness if it is accompanied by a rise on the inner corner of the eyebrows and by drooping eyelids, while it can convey disbelief if it is accompanied by a rise of the entire brow and a push up of the lower lip.

Although the internal state of a person can affect to the verbal content of the messages he/she conveys (for example, a question can be formulated in a harsher manner if the speaker is angry), affective information is mainly connected to the non-lexical component. Thus, the analysis will be directed towards determining the effect that affect states have over the non-verbal features of speech (prosody and paralanguage). This effect is going to be perceived primarily on the rate of the speech and its fundamental frequency, also known as pitch.

The effect that affect states have over the posture and motions of a person was already discussed by Darwin in *"The Expression of the Emotions in Man and Animals"* [139]. In 2009 [154], de Gelder stated that less of the 5% of works that research emotion in humans focuses on bodily expression, while 95% of them relied on the analysis of facial expressions. Kleinsmith and Bianchi-Berthouze [155] conducted a survey of the existing literature on affective body motions. As part of this survey, the authors analysed how the different works connected certain affect states to body motions. This review shows that, although some authors report similar body features associated to a given emotion, not all reports coincide.

The relationship between colour and emotion has been considered in diverse areas, from art to biology, or even language. For example, in English, *feeling blue* is used as a synonym for being sad, while in Spanish envy is connected to the colour green. Along the years, there have been several

researches aimed at finding the connection between colour and emotion. In [156], da Pos and Green-Armytage performed an study to connect facial expressions for basic emotions with colours. If we consider the position that defends the universality of the expressions for basic emotions, then an assumption could be made about how connecting colours to said expressions could indicate the relationship between colours and basic emotions. Valdez and Mehrabian [157] studied the emotional reactions elicited by colour hue, saturation, and brightness. While the two latter showed a strong and consistent effect, the same cannot be said for colour hue. Sutton and Altarriba [158] presented a study that had the aim to obtain a set of norms that could connect positive and negative emotions and emotional terms with colours.

4.4 Principles of the proposed approach for managing a robot's expressiveness

This section introduces the requisites and principles, both theoretical and technical, that have been defined for the development of an expressiveness management architecture for social robots. First, the design requisites that the final approach has to fulfil are described. Next, I justify the selection of state machine-like structures as the best solution for describing the expressions of the robot. Finally, the three proposed modulation techniques are introduced more in detail, although without going into technical details.

4.4.1 Design requisites for a robot's expressiveness system

In early stages of the work developed for this dissertation, a series of requisites were defined for the design of the new expressiveness architecture:

- **Easy design:** It is important that new expressions can be generated with ease and without requiring programming experience. The benefit is twofold: it reduces the time cost of creating expressions for new applications integrated in the robot, and allows developers to include people without a technical background into the design process.
- **Adaptability:** To overcome one of the problems of relying on a limited library of predefined expressions, it is desirable that each expression can be parametrized and modulated. That way, the expressions can be adapted to different situations , and used to convey the internal state of the robot.

- **Complexity:** While an expressiveness system could be developed using exclusively basic actions, like move joints, utter sentences, or change the expression of the eyes, having the option of integrate complex communicative skills can help to achieve a more realistic behaviour without adding complexity to the expression design process. An example of a communicative skill would be tracking and following the face of the other speaker.

- **Concurrency:** The need for handcrafting very complex expressions that combine multiple actions through several communicative channels can be avoided if that complex expression is modelled as a combination of several basic expressions. In the proposed HRI System, the most clear example is the combination of expressions coming from the Liveliness module, which have the goal of enhance the robot's animacy, and those coming from the HRI Manager, which aim at achieving specific communicative goals. Instead of having to combine both expressions into one, it is simpler to allow both to be executed independently. Thus, the proposed architecture needs to be able to perform multiple expressions at the same time.

- **Modularity:** A modular architecture simplifies the process of adding, removing or upgrading the different functionalities that the expressiveness system provides. It is important to find a balance between the benefits of having a distributed architecture and the possible drawbacks (mainly, the delay that can be generated by the communications between the different modules of the architecture).

- **Response time:** In Section 3.6.2.2, it was stated that, for communication to be successful, messages have to be conveyed in a specific period of time, or their meaning might be lost. Thus, the expressiveness architecture has to be able to execute expressions at a speed that meets the requirements for human communication.

- **Action Synchronization:** The proposed architecture should give freedom to the developers when combining unimodal actions (actions involving a single mode of communication) into expressions. This means that expressions should allow for an undetermined amount of actions to be performed, either sequentially or in parallel, and also should allow developers to tie the execution of these actions to either points in time, or to the completion of other actions.

The combination of these features will provide a multi-purpose expressiveness system that is easily extensible with new behaviours and adaptable to unforeseen circumstances and changes in the robot's internal state. Once the main characteristics of the new system were defined, the next challenge to solve was to decide how the expressions of the robot would be modelled.

4.4.2 Developing a model that represents multimodal expressions

The model used to represent the expressions of the robot is going to define how the individual actions of the robot have to be described, as well as how these actions connect with each other. When facing this task, two possibilities were considered: decouple the different communicative channels, or consider multimodality from the beginning of the process. With the first option, the actions that will be performed by each communicative interface are generated separately, and then synchronized afterwards by timing appropriately the start of each unimodal action sequence. The second option allows to connect actions from different interfaces directly in the model, providing a more precise synchronization between all the modalities, instead of relying exclusively on timing. This is the approach followed in this thesis.

In previous versions of the expressiveness architecture of the robots used in this thesis, expressions were modelled after music scores. Each expression contains a list of multimodal actions, and the points in time in which each action has to be performed. While this approach allows to consider multimodality during the design of the expressions, it has the limitation of having to use time points as the only synchronization mechanism. This presents two potential drawbacks. On one hand, it forces developers to measure manually the length of each action, and usually rely on trial and error to ensure that the end result of the synchronization is the expected. On the other hand, if the execution of one or more actions deviates from the ideal (for example, if the initial position of a joint is not the one expected, which would lead to a longer or shorter motion), then the timing of the whole expression would be off, which could lead to the expression losing its meaning, or at least not having the same effect. The solution to both problems was to allow the connection of the end of one action and the beginning of the next one. This would eliminate the need for measuring the length of the actions, and at the same time would take into account any unexpected delays in the performance of actions. While this synchronization approach is easy to implement for unimodal expressions, it becomes more of a challenge when multiple types of actions are involved. Thus, an appropriate structure that would allow to represent these connections between actions had to be chosen.

After considering different possibilities, a decision decision was made to use state machine-like structures to model multimodal expressions. Under this approach, states represent the actions integrated in the expression, while the transitions between states take care of the synchronization. States in this model can be categorized in one of two types: *action* states and *flow management* states. The former are used to command the different interfaces of the robot, while the latter are used to control the execution flow of the expression. On top of controlling action states, flow management states also allow to integrate in the expression design structures taken directly from a computer

science background, like loops or branching structures. Loops help to simplify the creation of new expressions, as developers can generate repetitive motions by specifying only one motion cycle and the number of repetitions. A case of use for this feature would be the design of a waving expression with only two arm movements that are repeated multiple times. Branching, on the other hand, can be used to design multiple action sequences for a single expression, and then select one of the sequences depending on context. For example, a greeting expression could be design with different actions depending on the time of the day, or the type of user. Without branching, this would have to be solved by creating two expressions that could be almost identical with only slight variations, which is time consuming and makes the library of expressions less manageable.

While traditional state machines have the constraint that only one state can be active at the time, the proposed structure for modelling expressions allows for concurrent execution of states, as a mean for performing multiple actions at the same time. With this, developers have the freedom of selecting how much actions should depend on each other. An expression could choose to separate actions from different interfaces in individual branches in the state machine that are executed concurrently, and use only timing as a synchronization strategy. This not only gives developers more tools for expression design, but it also allowed to port all the expressions developed for the previous version of the expressiveness architecture without changing their structure. Finally, is important to mention that *action* states not only perform basic unimodal actions, but can be used to activate and deactivate complex communicative skills. Continuing with the face tracking example given in the previous subsection, an expression could activate the face tracking skill at some point, wait for a while, and then deactivate it whenever is no longer required, without having to worry about how the face tracking actually works. Finally, as a way to increase the modularity of the proposed approach, expressions can be nested to create more complex behaviours. This allows developers to model combinations of actions that appear in multiple expressions as a single gesture, and then create as many expressions as needed by just integrating this gesture as a new state in the state machines representing the expressions.

4.4.3 Expression of internal states in a social robot

Having a model that can be used to represent the multimodal expressions of the robot is not enough to guarantee a satisfactory interaction. Based on the knowledge discussed earlier in this section, it would be beneficial that the robot's expressions not only convey the communicative intention (e.g. greeting a person, or asking a question), but also transmit the internal state of the robot. This can help to make the expressiveness of the robot to look more human-like, and could facilitate that the user grows more attached to the robot, improving their interactions, and thus helping the

robot to perform its role better. Although there are different approaches for solving this particular problem, this dissertation proposes the use of different modulation techniques for adapting the robot's expressiveness to its internal state.

Three different modulation strategies have been considered in this dissertation:

1. **Dynamic reconfiguration:** With this technique, actions in a expression can be changed dynamically. For example, the default version of a greeting expression could make the robot wave its arm and utter the sentence *"Hello, how are you?"*. An application could request the performance of that expression, but replacing the sentence with *"Good morning, how are you?"* in order to reflect the moment of the day. A variation of this strategy allows developers to use keywords to mark points of the speech where information from the context should be inserted. All keywords start with the @ character, and are directly mapped to information stored in the memory of the robot (e.g. the @name keyword would be replaced with the name of the user the robot is interacting with). If a keyword does not return a value, then the system simply removes it.

2. **Parameter-based modulation:** This strategy allows to modify the global appearance of an expression based on two dimensions: *speed* and *amplitude*. The effect that each of this variables has is tied to specific parameters of each communicative interface. *Speed* affects the velocity of the motions, the speech rate, the blink frequency of the eyes and also the LED, while *amplitude* modifies the volume and pitch of the voice and the brightness of the LED. In this strategy, both speed and amplitude are considered as discrete variables with seven possible values: low, medium, and high increase and decrease, as well as a neutral state that will be used as the default value.

3. **Profile-based modulation:** The last modulation strategy allows developers to define how the state of the robot will affect to each communicative mode individually. In the proposed expressiveness approach, each interface is controlled by a series of parameters that can be used to generate actions (for example, for the voice, these parameters include the pitch, the prosody rate, the volume, the sentence that will be uttered, etc...). Developers can create modulation profiles that specify how each possible state of the robot affects to these parameters (e.g. if the robot is happy, the pitch of the voice and the speed of the motions should be increased). However, giving developers total freedom about what values to use for each parameter has two main drawbacks. First, it is not very intuitive what effect these values have. It might be difficult for a developer to know if increasing the speed of the motors from 1.5 to 2.75 (for example) is an excessive change or not. Second, an excessively aggressive modulation could result in the

expressiveness of the robot being unnatural (for example, raising the pitch above what would be normal for a human). In order to avoid these drawbacks, the profiles specify the values for all the interfaces' parameters as a percentage of a range between two limit values, which have been selected to ensure that the actions are still human-like. When the internal state of the robot changes, the expressiveness architecture takes the new values from the modulation profiles, and uses them to adapt any expression being performed.

The details of how these strategies were implemented will be presented in the next section of this dissertation.

4.5 Implementation of the Expression Manager

This section presents the Expression Manager, the element in the software architecture that controls all of the robot's expressiveness capabilities. It manages the actuators of the robot when they are used in communicative tasks, according to the requests coming from other modules in the architecture. Although any module can request the use of one or more output interfaces, the majority of requests come from either the HRI Manager (the dialogue manager introduced in Chapter 3) or the Liveliness module (see Chapter 5). Based on the requests received, the Expression Manager loads the appropriate expressions from the gesture library, and ensures their correct execution. Besides controlling the loading and execution of expressions, the Expression Manager has to protect the system against conflicts derived from two applications trying to use the same interfaces at the same time. This is done through the prioritization of some expressions over others.

In the proposed software architecture, the Expression Manager is modelled as a combination of several processes, and is in charge of the following four tasks:

- **Planning of expressiveness:** This is the first stage in the expressiveness pipeline. The Expression Manager evaluates what interfaces the requested expression and the gestures being performed (if any) require, and decides if and when the requested expression has to be performed.
- **Execution of expressions:** Gesture requests are stored in priority queues until they have to be executed.
- **Internal State expression:** The Expression Manager is in charge of conveying the internal state of the robot through a proper modulation of the gestures.

- **Communication with the output interfaces:** The Expression Manager creates a bridge between the software architecture and the robot's actuators when they are used for interacting with users. For example, if the robot's motors have to be used to greet the user by waving a hand, this command is sent through the Expression Manager. However, if the motors are used to navigate the environment (which does not involve communication with an user), then the Expression Manager is not involved in relaying these commands.

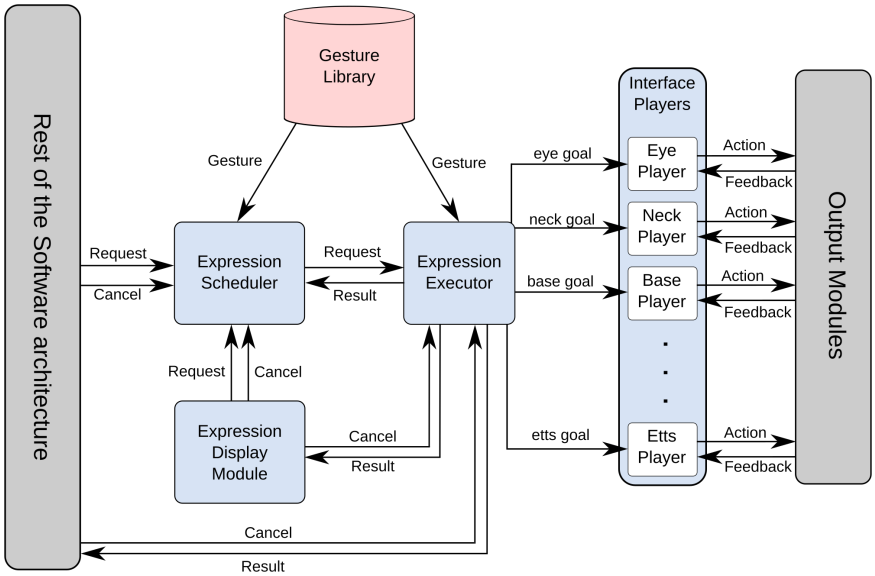


Figure 4.2: Software architecture of the Expression Manager.

Structure-wise, the Expression Manager is divided in a series of modules that can share information between them, as shown in Figure 4.2. These modules are (i) the Expression Scheduler, (ii) the Expression Executor, and (iii) the Interface Players (there is one player for each output channel). Figure 4.3 shows the class diagram for the Expression Manager. The Expression Scheduler, Expression Executor, and each Player have been programmed as individual classes that interact with each other through the infrastructure provided by ROS. In a robotic platform, individual instances of the JointPlayer class exist for each joint. Expressions have been modelled as individual classes that inherit from a common template: the BehaviourTemplate. Each expression can be composed of one or more states, each modelled as an individual class. Finally, while the Expression Scheduler loads the expressions directly by instantiating an object from the expression's class, the Executor instantiates first an object from the GestureSM class, which provides the features required for running expressions in parallel execution threads, while this object in turn loads the expression. All the elements and processes mentioned here will be presented in depth in the following sections.

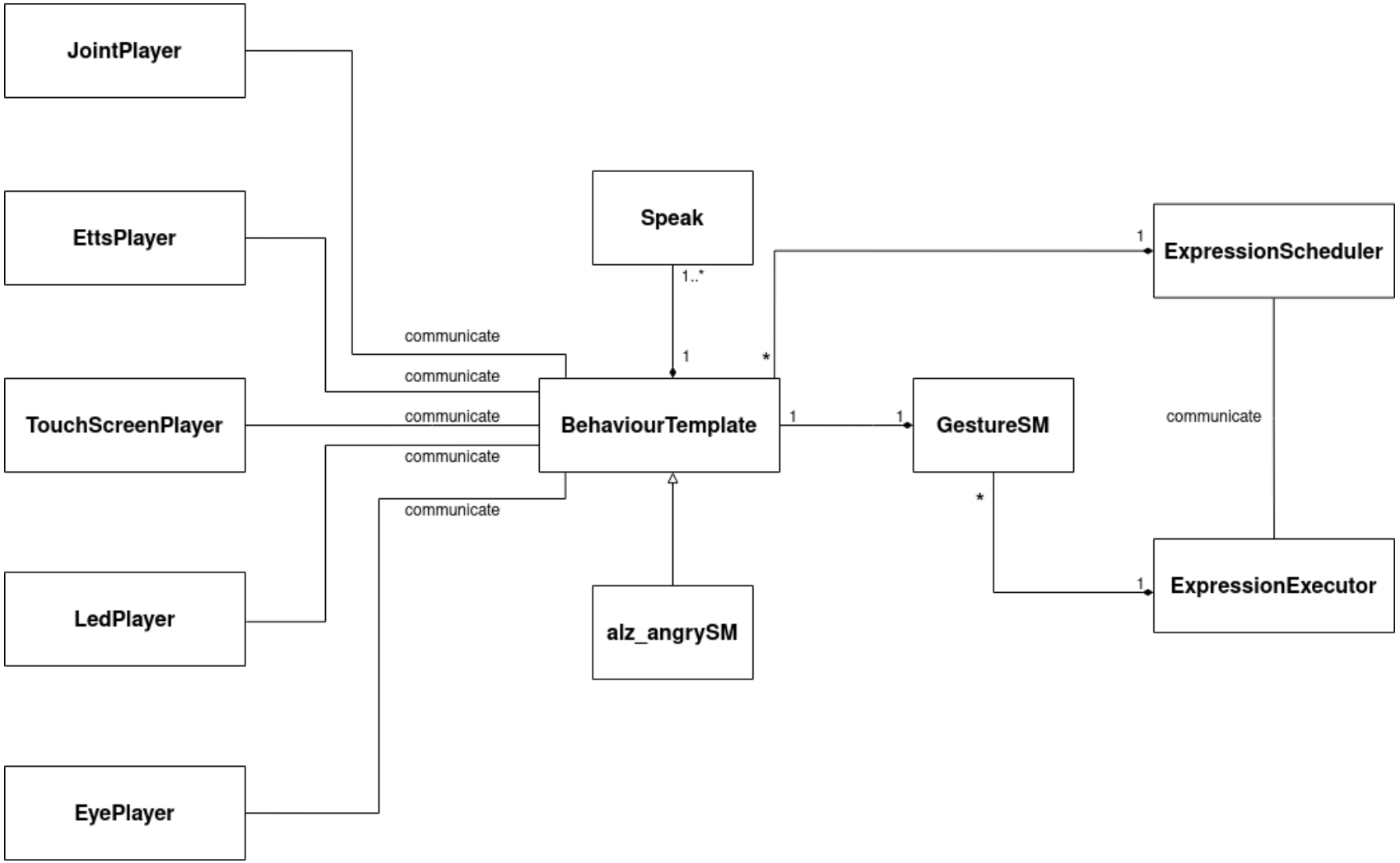


Figure 4.3: Class diagram representing the relationships between the modules of the Expression Manager and the library of gestures. The list of attributes and methods for each class can be found in Appendix B.

4.5.1 Software Modules

The first module in the Expression Manager is the Expression Scheduler. This module processes all the requests for gesture execution coming from the rest of the architecture, checks the current status of the robot's interfaces (if they are being used or not) and matches this against the list of communication modes that the requested gesture needs. Based on this analysis and on the priority assigned to each expression, the Expression Scheduler can decide if the new gesture has to be executed (at the cost of stopping any conflicting expression), queued, or discarded.

This module serves as the connection between the Expression Manager and the remaining modules in the robot's software architecture. Expressions are requested through a standard message containing all the information required for loading and parametrizing the appropriate expression. The parameters included in this request are:

- **name:** indicates which expression has to be loaded from the library.
- **id:** a unique string used to identify the expression. This will be useful in situations where multiple instances of the same expression are active at the same time (this is a situation that can arise for very specific gestures).
- **emitter:** The module in the architecture that requests the activation of the expression.
- **priority:** Priority level that will be used to solve any conflict between two expressions regarding the use of the output interfaces. Three levels can be assigned: low, medium, and high.
- **params:** Key-Value Pair Array that allows applications to specify changes that have to be applied to the requested expression.

There is a general rule for assigning priorities to the expressions. In general, gesture requests coming from either the HRI Manager or the applications will be assigned a medium priority. This will be considered the default value for any gesture request. Applications can send high priority requests whenever they require that the robot displays an expression immediately, regardless of what other actions the robot is performing. Usually, high priority gestures are used in situations where the robot needs to react immediately to an external event. An example of this would be the robot reacting to a loud noise. Finally, low priority is usually given to gestures that do not play a key role in achieving communicative goals or reacting to events, but instead complement expressions with mid or high priority, as a way to enhance the quality of the robot's expressiveness. Primarily, this level of priority is reserved for the expressions requested by the Liveliness module.

Once the Expression Scheduler decides that an expression can be executed, then it sends an activation request to the module in the Expression Manager in charge of activating these gestures: the Expression Executor. This module receives activation requests coming from the Expression Scheduler for loading and executing gestures (the message used for the request is the same one described for the Expression Scheduler). While the Scheduler controls that there are no conflicts between expressions and checks if a request can be processed or not, the Executor controls how the requested gestures are loaded from the library, parametrised, and then executed. The activation orders sent by the Scheduler are stored in priority queues, one for each priority level. At a certain rate, the Executor extracts gestures from the queues, following a first-in first-out approach. While the loading and configuration of these gestures is done one by one, they are executed in different threads, so the system can have multiple expressions active at the same time. Also, any cancellation request received in the Expression Scheduler is then relayed to the Executor, which in turn sends the signal required to stop the expression.

The Interface Players (from now on, simply Players) are the modules that connect the expressiveness architecture with the output interfaces of the robot. In the proposed system, there is an individual Player for each communication channel, resulting on a total of 10 modules: one for each joint (in Mini: head, neck, both arms, and base. The base player is not present in Gero), two for the LED (heart and cheeks), one for the TTS, another for the eyes, and a last one for the touch screen.

All players have been designed using a common template based on ActionLib [1], a library included in the ROS framework that defines a standard interface between a client and a task. In this sense, it is similar to how a service works in ROS, but with a particularity: ActionLib servers can provide feedback regarding the status of the task, and also have a mechanism that allows the client to cancel the current task at any given time. While the task performed is different for each Player, the use of ActionLib provides three features shared by all Players:

- At any given time, the Expression Manager can enable or disable a Player to adapt the interaction to different situations. For example, in very noisy environments, speech-based communication might be highly inefficient, and the TTS Player might be deactivated, switching to other communication strategies.

- Regardless of the interface they are managing, all Players have strategies in place for controlling the correct execution of their task. They will provide feedback about the state of the action being performed, and notify the result once it has ended.

[1] `https://wiki.ros.org/actionlib`

- An action in course can be interrupted at any time. This will be used primarily for stopping the execution of expressions if a new gesture with a higher priority is received.

In the proposed architecture, the Players receive actions exclusively from the robot's expressions. Also, because no interface can perform two actions at the same time (for example, the TTS cannot utter two sentences at the same time), the Players will only manage one action at a time. When a Player receives a request for executing an action, it extracts from the ActionLib goal all the information that will be used to configure the message that has to be sent to the output interface. This message is then completed with information that is stored in the context of the robot. For example, an application can include the keywords @name and @robot in utterances sent to the TTS Player, which would replace these keywords with the names of the user and the robot, respectively. After the message for the output interface has been properly built and sent, the Player keeps track of the requested action's status through the feedback provided by the interface, and relays this feedback to the gesture that requested the action. Players also integrate safety mechanisms that isolate the HRI Architecture for any unexpected failures in the output interfaces. These failures include the loss of communication with the interface, or an unexpected cancellation of the action (for example, a motor stopping before the final position is reached) among others. Once the action has been completed (successfully or not), the Player sends the result of the task to the expression that requested the goal originally.

4.5.2 Modelling gestures

As already mentioned, the gestures used by the Expression Manager are modelled as state machine-like structures. This opens the possibility of using design strategies that do not require manually coding the expressions. Instead, the goal was to use a graphical interface to create new gestures, as a mean to allow developers that do not have a background on computer science, but possess an expertise on other areas (for example, in animation), to design gestures for our robots.

Multiple options were considered when evaluating the possible tools that could be used for creating gestures, including the possibility of designing this tool from scratch. In the end, a decision was made to use FlexBE [159], a ROS-based framework for designing high level behaviours for robotic applications. It was built as a layer on top of SMACH, the same library that was used in the creation of the CAs described in Chapter 3. FlexBE states and state machines inherit from SMACH, adding new features on top of the ones already provided by SMACH states and state machines. Besides expanding the features provided by SMACH, FlexBE also provides a Graphical User Interface (GUI) that can be used to design new behaviours by connecting blocks that represent

states, and also allows the roboticist to teleoperate the robot, giving feedback in real time on the status of the behaviour being run. Figure 4.4 shows an example of a basic gesture created with FlexBE's GUI.

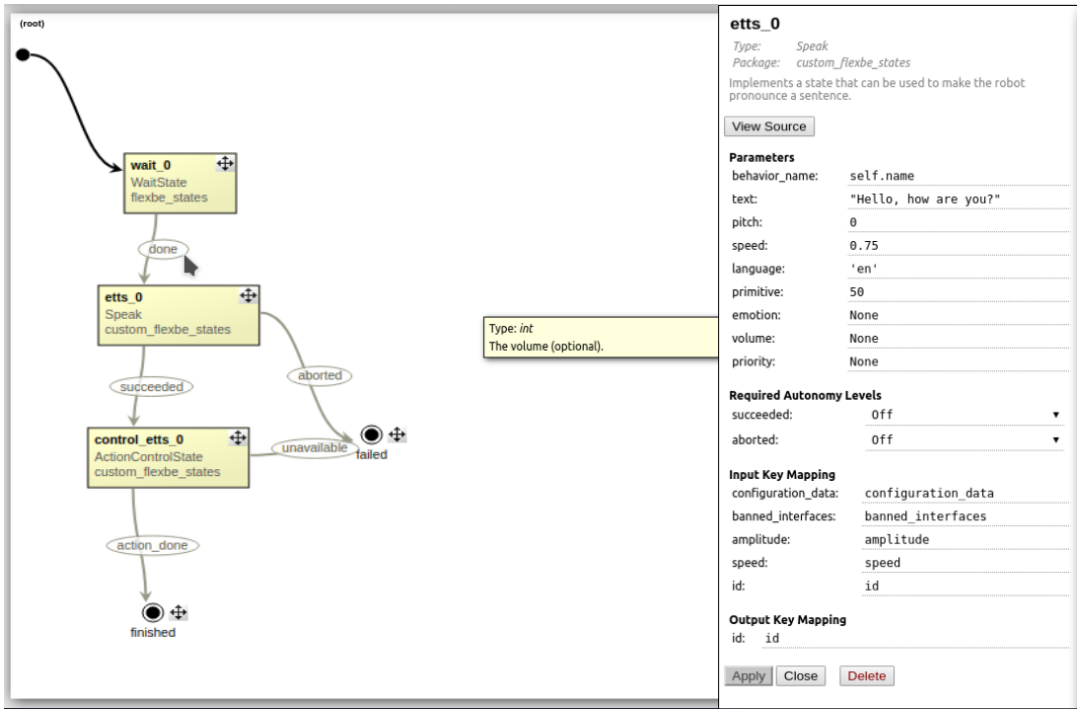


Figure 4.4: Example of a basic gesture created with FlexBE. This gesture waits for 5 seconds, then utters a sentence, and waits for the sentence to end.

Four modifications were applied to FlexBE in order to adapt it to the needs of the expressiveness architecture proposed in this chapter. First, FlexBE's source code had to be altered to account for multiple state machines running in parallel. Second, the initialization of FlexBE state machines was optimized by removing every feature that will not be used by the expressions. Third, the FlexBE GUI was modified to automate the initialization of all the parameters common to all gestures. Finally, a standard template was designed, from which all the expressions created inherit. This template implements the functionalities required for finalizing the execution and returning the result of the expression.

While FlexBE provides a library of states that can be used to create behaviours, a new library was designed for the purpose of designing robotic expressions. This library includes individual states that are used to communicate with each Interface Player. These states receive as inputs all the parameters that define the action that has to be performed and use them to build the goal that has to be sent to the corresponding Player. The library also include states that are in charge of controlling the gestures' flow

of execution. Table 4.2 shows a description of all the states developed. A more in depth description of each state, along with the parameters that can be used to configure them, can be seen in Appendix C.

Under FlexBE, states in a state machine are grouped using a series of containers that provide different functionalities. In particular, there are two that play an important role in the creation of expressions: *sequential* and *concurrent* containers. *Sequential* containers allow for a lineal execution of a sequence of states, while *concurrent* containers control multiple states that have to be executed at the same time. These containers can be combined and nested to create structures that represent expressions that are as complex as required. Developers load the states and containers as blocks in FlexBE's GUI, parametrize them accordingly, and then connect them to define how the expression has to be executed.

State	Description
Action Control	Waits for the completion of an action sent to a Player. This state can wait for multiple actions.
Color LED	State used to send commands to the LED Player.
Display Touch Screen	State used to send commands to the Touch Screen Player.
Express Eyes	State used to send commands to the Eyes Player.
For Loop	State used to implement a for loop. It keeps the count of the iterations and returns a different outcome depending on if the final iteration has been completed.
If Loop	State used to implement an if loop. It checks the value of a parameter and returns a different outcome depending on this value.
Move Joint	State used to send commands to the Joint Players.
Reset Interfaces	State used to reset all the robot's interfaces to a default value.
Return Result	State used to return the result of the expression (if the execution was successful or not).
Select Random Gesture	State that selects a gesture randomly from a list.

Table 4.2: List describing all the states that have been developed for designing expressions.

State	Description
Execute Random Gesture	State that executes the gesture selected by the Select Random Gesture.
Start Skill	State used to activate complex skills.
Stop Skill	State used to deactivate complex skills.
Speak	State used to send commands to the ETTS Player.

Table 4.2: List describing all the states that have been developed for designing expressions.

4.5.3 Executing expressions

The process of executing an expression is a distributed task that involves all the modules in the Expression Manager. Requests sent by the HRI Manager or other modules in the architecture are managed by the Expression Scheduler. When one of this requests is received, the Scheduler has to load the gesture from the library and extract the list of interfaces that will be required. Next, this list is matched against the interfaces that the currently active gestures need. This does not only include the interfaces in use whenever the request is received, but any interface that the gestures are going to use during their whole execution. If a coincidence is found, then the priorities of the new and the currently active gestures are compared. Four results can be obtained from this comparison:

1. If the new gesture has low priority, then it is discarded, regardless of the priority level that the active expression has.

2. If the new gesture has at least mid priority level, and the active gesture has either the same or a higher priority, then the new expression is stored in the proper priority queue and executed whenever the interfaces are available.

3. If the new gesture has either mid or high priority and active gesture has low priority, then the new expression is executed immediately, while the active gesture is stopped and discarded.

4. If the new gesture has a higher priority, but the active expression has mid priority, then the new expression is immediately executed and the active one is stopped, stored in the priority queue, and executed again whenever the interfaces are free.

From these situations, a series of conclusions can be extracted. First, high priority gestures cannot be interrupted by anything new. The reason behind this decision is that if an expression is

important enough to require high priority, it should always be executed completely. Second, low priority expressions will be completely discarded if the interfaces are being used. This means that only non-essential gestures that lose their meaning if they are not performed immediately should be requested with low priority. Finally, the last conclusion is that anything that the robot is currently performing is going to be more important than any new expression, if the priorities are the same. This is done because it was considered preferable that the robot finishes whatever is doing before jumping to something new, instead of just interrupting actions mid-execution, because this could be perceived as strange by the user.

If either no conflicts arise, or the outcome of the comparison between the new and the active gestures lead to the new expression to be immediately executed, then the Expression Scheduler relays the activation request to the Expression Executor. At the same time, the Scheduler generates a watchdog timer with the goal of ensuring that, in the event of an unrecoverable error, the expression does not get stuck, blocking the access to the output interfaces. The request activation is then stored in the corresponding priority queue. The Executor extracts the requests one by one, loads the appropriate template from the gesture library, configures it with any extra parameter contained in the request activation, and runs the expression in a parallel program thread, in order to achieve concurrent, asynchronous execution of gestures. Whenever the gesture is completed, it sends a message containing the name and ID of the expression, a string that indicates if the execution was successful, failed, or was cancelled, and, if it failed, an optional field that might indicate the error encountered. This result is received at the same time by both the Expression Scheduler and the Executor, as well as the module of the software architecture that requested the gesture in the first place. After removing the completed expression from the list of active gestures, the Scheduler checks every gesture stored in the priority queues to see which of them were waiting for the interfaces that were just freed. The gestures found are removed from the priority queues and sent to the Executor. This process continues until all priority queues have been emptied.

Gestures can be cancelled at any given time. These cancellation requests are sent to the Scheduler and the Executor. The former removes gestures that are scheduled to be executed, while the latter stops expressions already in execution by sending them a termination command. Once the gesture has been cancelled, the process described in the previous paragraph for extracting gestures from the priority queues and execute them is repeated. Finally, if at any time one of the expressions fail, leading to the watchdog timer going off, then the Scheduler forces its cancellation and sends a deactivation command to the Executor.

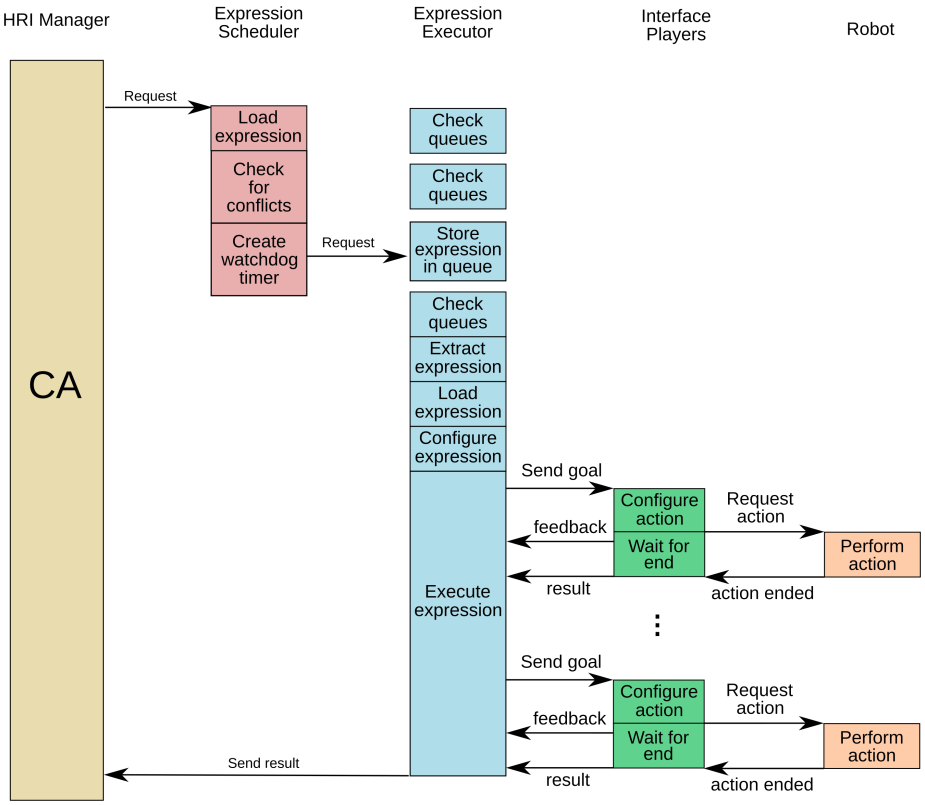


Figure 4.5: Activity diagram depicting all the steps performed by every module involved in the execution of expressions.

4.5.4 Modulating Expressions

The modulation of the robot's expressions is, by most part, managed internally by the Expression Manager. In this dissertation, three different modulation strategies were considered: (i) state-based customization; (ii) global modulation; and (iii) profile-based modulation.

The state-based customization method allows to modify expressions in runtime by changing the parameters that define an action. Any parameter can be overwritten (e.g. the speed of the motions, or the sentence that the robot will utter). The new configuration for the expression's actions will be included in the request that the Expression Manager receives for executing the gesture. The request will also indicate to which state the changes have to be applied. For example, if the request includes changes to a utterance, but the expression includes two different utterances in separate states, the request has to indicate if it is the first one or the second one the action that has to be modified. The following example can help to illustrate the utility of this modulation approach. The library of gestures used by the robot includes an expression for greeting users, where the robot waves its arm

and says *"Hello"*. This gesture can be used under any situation, but because of this, it can also be too generic. A possible solution could be to alter the sentence and adapt it to the circumstances of the interaction (for example, using a greeting that is specific for the user approaching the robot).

If the activation request for a gesture has any extra action in it, the Expression Executor passes the new list of actions to the gesture during the configuration stage, before executing it. Each action is tied to a specific state name. This list is sent to all the states in the expression, which in turn check if their name is on the list. If it is, then the state extracts the new configuration for the action and overwrites the default one. While this strategy is useful when an application needs to modify specific parts of an expression, it has two main limitations. First, the overall structure of the expression has to be maintained. This means that no new actions can be added, and none of the existing actions can be removed. So, in the previous example about the greeting gesture, an application could modify the arm waving or the utterance spoken, but neither remove the arm waving nor add a head motion. Second, this method requires that each action in an expression is modified individually. Thus, if a change has to be applied to multiple actions (for example, increase the speed of all the motions in the expression), this would force to specify the same change for all the states, which can be time consuming (specially if the expression has a high number of states).

	Speed	**Amplitude**
Voice	Prosody rate	Volume and pitch
Joints	Motion speed	None
LED	Fading speed	Brightness
Eyes	Blinking speed	None

Table 4.3: Effects that the modulation parameters have on each type of interface.

The global modulation method aims at solving this last limitation by providing control parameters that allow applications to regulate the global behaviour of an expression. The two parameters selected are *speed* and *amplitude*. The first parameter affects to the temporal dimension of the actions, while the second one modifies the intensity of the expression. Both control parameters have different effects for each of the interfaces, as shown in Table 4.3. On a first approach, *speed* and *amplitude* were developed as scaling factors that were used as multipliers for the affected parameters in each communicative action. But this solution was deemed to be too unintuitive, as it is fairly hard to evaluate which effect an increase on speed of a 1.5 translates to with regards to the speed of the motions, for example. Thus, the modulation of the control parameters was

discretized into a finite amount of intervals: high decrease, medium decrease, low decrease, low increase, medium increase, and high increase. A seventh value, normal, is used as the default configuration for the expressions. Limit values were empirically determined for each of the parameters presented on Table 4.3. These values do not represent the actual limits imposed by the hardware (for example, the highest speed value that can be sent to the motor), but instead a threshold value past which the action might no longer be perceived as natural. For example, while the TTS of the robot allows to increase the pitch past the limit set, the voice that result from using those pitch values stops resembling the voice of a person, and instead is more similar to that of a cartoon character.

Applications can include one of the six values for both control parameters in the activation request for a gesture. When the expression is configured, the values for *speed* and *amplitude* are sent to all the states. During the execution of a state that has to send an action to the Players, the following steps are followed. First, the state retrieves the limit values for the parameters shown in Table 4.3. If the *speed/amplitude* has to be increased, then the top limit is retrieved. Otherwise, the bottom limit is retrieved. Second, the modulation for each parameter is computed computed using Equation 4.1:

$$V_f = V_i + s * (V_l - V_i) \tag{4.1}$$

In this equation, V_f is the value after the modulation was applied, V_i is the default value of the parameter, V_l is the limit value selected (maximum or minimum), and s is the size of the step taken during the modulation. It will take the value 0.33 for low increases and decreases, 0.66 for medium increases and decreases, while for high variations it will take the value 1, which means that the final value will be the exact limit value.

As stated in Section 4.1.2, human communication involves two components: the symbolic component seeks to convey a particular communicative goal, while the spontaneous dimension conveys the internal state of the speaker. Thanks to this modulation approach, expressions designed with only the symbolic component in mind can be adapted to represent also the robot's state. For example, if the robot has to look enthusiastic, this would translate in a change in multiple aspects of its expressiveness (higher speed and amplitude of motions, prosody rate increased, higher voice volume, etc...). This state could be achieved by modulating the robot's expressions with both high *speed* and *amplitude*. But this solution still presents some limitations. From a theoretical point of view, the suggested modulation strategy has the problem of tying the variations of all the communication channels together. This makes it impossible to modulate a specific characteristic of an expression without changing the rest of them. For example, one cannot use the *speed* control parameter to change the speed of the robot's motions without forcing a similar modification of

the voice's prosody rate. From a practical point of view, this modulation strategy forces developers to specify values for every single parameter in an action when designing expressions. Currently, interfaces are able to assign default values to any parameter that is not defined in the gestures, allowing developers to focus only on those parameters that are important to the gesture. For example, when creating a motion, developers can define only the joint positions, while the speed and acceleration are selected by the software that controls the motors. But if a parameter is not defined in an expression, then it cannot be modulated using the *speech* and *amplitude* parameters.

The profile-based modulation approach strategy developed corrects both the theoretical and practical issues presented in the previous paragraph. It is based on the creation of modulation profiles in which developers can define how a change in the robot's internal state should affect to each parameter in the robot's interfaces. The basic idea is similar to the one followed in the previous technique: the modulation that has to be applied to each parameter is defined as a variation in the possible range of values that the parameter can take (this range is defined as the difference between the top and bottom limits used in the global modulation method). But while *speed* and *amplitude* affect to the parameters that were defined during the design of the expression, the modulation profiles work the other way around, and only change those parameters that were left undefined. Thus, developers can focus on defining the elements of the expressions that they consider critical to transmit the meaning of a gesture, while the Expression Manager can use the remaining parameters to convey different internal states. In the modulation profiles, the effects are described as the percentage of the range between the limit values for each parameter. For example, if the pitch of the voice can take values ranging from -5.0 to 10.0, then a value of 0 will be interpreted as the 0% of this interval (-5.0), a value of 100 will be transformed into the top limit (10.0) and a value of 33.3 will be transformed into the value 0 (a third of the range). For each profile, developers can define effects for multiple internal states. Currently, this approach has been tested for conveying affect states, both emotions and moods. The mood profile includes values for five possible states (neutral, happy, anxious, bored, and calm), and the emotion profile considers another five possible states (neutral, happy, sad, angry, and surprised). Figure 4.6 shows the values defined for the *happy* state, extracted from the emotion profile.

This modulation strategy is implemented directly in the Interface Players. During the startup of the Expression Manager, the Players load all the available modulation profiles and store the effects that all possible internal states can have over the parameters related to the interface controlled by each particular Player. The *neutral* state is considered to be the initial one, and it is stored as the active configuration. If a change in the internal state of the robot occurs, the Players upload the active configuration with the modulation tied to the new state (for example, a change in the robot's mood

```
'happy': {
    eyes:{
        expression: '14',
        blink_frequency: '50',
        x_position: 50,
        y_position: 50,
        pupil_size: '50'
    },
    etts:{
        pitch: '133',
        speed: '77.5',
        volume: '55'
    },
    joints:{
        velocity: '40',
        acceleration: '40',
        head_initial_position: '-0.15',
        neck_initial_position: '0',
        leftArm_initial_position: '-0.2',
        rightArm_initial_position: '0.2',
        base_initial_position: '0'
    },
    heart:{
        color: '0xFFAF00',
        max_intensity: '50',
        fade_velocity: '65'
    }
},
```

Figure 4.6: Example of the modulation values for the *happy* state in the emotions profile.

would lead the Players to load the modulation values that correspond to the new mood). Whenever an expression sends a goal to one of the Players, it goes through all the parameters that can be sent to the output interface. If a parameter is defined in the goal, then the Player uses the value sent by the expression. On the other hand, for every parameter that does not have a value defined in the goal, the Player checks the active configuration and selects the appropriate value.

4.5.4.1 Customizable expression

As discussed before, one of the disadvantages of using a finite amount of expressions is that interactions tend to become repetitive if the size of the library is not big enough, and also that is time consuming to create expressions that represent all possible internal states of the robot. While the modulation techniques presented in the previous section can help alleviate this problem, it is still desirable to make the expressiveness system as adaptable as possible. In order to cover any potential situation not considered during the creation of the gesture library, a generic FlexBE template was developed for generating an expression dynamically. This template can be requested through the same process followed for any other gesture, with a request containing the list of actions that have to be used to create the new gesture. Once the gesture is sent to the Expression Executer to be loaded, the list of actions is sent to the customizable gesture template, and the state machine is built on the fly. This sequence of actions is represented as a dictionary, where the values are all the parameters required

to configure each action and the keys are a series of strings used to uniquely identify each action. The configuration for each action can also include two extra parameters: (i) a start time for that particular action (measured as a delay in seconds from the beginning of the gesture's execution), and (ii) the ID/s of the action/s that have to be completed before the execution of that action. All actions that do not specify either parameter will be performed concurrently at the beginning of the expression.

The creation of the customized expression is done as a two stage process. First, the template generates a tree that represents how the different actions relate to each other (when each action has to be performed, which actions have to be executed concurrently, or which actions have to wait for a previous one to be completed). Once this tree has been built, it is used as a blueprint for creating the corresponding state machine. The following example can help to illustrate the process of creating these dynamic expressions. In this example, the robot needs to perform a greeting gesture by raising and lowering the arm, and at the same time utter the sentence *"Hello! Nice to meet you"*. The activation request sent to the Expression Manager would then include three actions: (i) the utterance; (ii) the motion to raise the arm; and (iii) the motion for lowering the arm. The first two are independent, while the latter is connected to the completion of the previous motion. The resulting state machine would be modelled as a *concurrent* FlexBE container, with two *sequential* containers inside. One is used to utter the sentence, while the other includes the sequence for raising and then lowering the arm.

This customizable expression allows to create new gestures on the fly. Thus, expressions can now be designed manually and stored in the gesture library, can be obtained through the proper modulation of a predefined expression, or can be built from a set of individual, unimodal actions in runtime.

4.5.5 Emotion display module

Section 4.3.4 presented a theoretical basis of the expression of affect states in humans. The evidence presented in this section suggested that, according to some authors, basic emotions tend to be recognized universally (although they do present small variations between different groups of people). The section also presented some of the effects that affect states can have over the different communicative channels of a person (e.g., facial expressions, body language, speech patterns...). While adapting these features might be enough for users to recognize the affective state of the robot, some times it is also important that the robot shows a specific reaction to the stimulus that caused the change in the affect state, instead of just displaying a generic affect expression.

In the proposed system, this was achieved with the development of an Emotion Display Module that generates proper reactions to stimuli received. Figure 4.2 shows how the Emotion Display Module is connected with the Expression Manager. This module has been programmed to show reactions to a change in the dominant emotion. Using the software architecture in which the work developed in this thesis has been integrated, the robot can convey four different emotions (happiness, anger, sadness, and surprise), with an intensity value ranging from 0 to 100 in a continuous scale. At any given time, the robot can be feeling more than one emotion, with different intensities. In this context, the dominant emotion is the one with the highest intensity.

Whenever an emotion is triggered with an intensity level higher than a fixed threshold (in the current implementation, this threshold was set to 80), the Emotion Display Module reacts by requesting the execution of a gesture that conveys an appropriate display of that particular emotion. This process is not only affected by the emotion that has been triggered and its intensity level, but also the stimuli that triggered that emotion. For example, while there are multiple reasons that can cause the robot to be angry, the reaction might not be the same if the robot was simply treated with disrespect, or if the user hit it. In order to account for this distinction, the Emotion Display Module loads during its initialization a configuration file that contains which expressions have to be requested given an emotion and an elicitor.

Whenever the Emotion Display Module receives a new message from the robot's affect generation, first it has to extract the dominant emotion, its intensity, and the elicitor that triggered it from the message. These parameters are used to retrieve the list of expressions that can be used as a reaction to the emotion being triggered. The result of this search is an string made by the names of all the possible expressions separated by a delimiter character. The module selects one of the expressions at random and sends it to the Expression Manager to be performed. In this case, the emotional reactions are requested with high priority, as they are behaviours that should override whatever the robot is doing, and be executed immediately (Reactive behaviours can lose their meaning if they are not performed immediately after the stimulus that triggered them). If the module receives a new update of the robot's emotional state while the previous expression request is still being executed, then this update is ignored. Updates will be discarded until the Expression Manager notifies that the current emotional gesture has been completed.

4.6 Evaluating the Expression Manager

This section presents the experiments conducted to evaluate the effectiveness of the expressiveness architecture presented in the previous section. First, an objective evaluation will show the level of efficiency of the proposed expressiveness architecture, regarding the amount of resources consumed when the Expression Manager is integrated in a robotic platform. Next, two subjective evaluations have been conducted in order to understand how some of the capabilities of the Expression Manager affect the perception that the user has of the robot. In particular, these evaluations will focus on the effects of modulating the robot's expressiveness in order to adapt them to the context of the interaction.

4.6.1 Objective evaluation: Response time and resource usage

The objective evaluation has the goal of demonstrating that the proposed expressiveness architecture is able to work under real conditions, integrated in a robotic platform where a full software architecture is deployed. Under these conditions, the Expression Manager needs to ensure two features: (i) being efficient enough so it does not hoard a large portion of the available resources; and (ii) being fast enough to send the appropriate commands to the output communication channels without hindering the quality of the interaction, regarding its naturalness. Similar to the HRI Manager, the Expression Manager can be considered as another of the core modules in the robot's architecture, thus making these features that much important.

This evaluation has been performed on Mini, under the same conditions used for the evaluation of the HRI Manager in Section 3.6.2.1 (the robot had the entire software architecture running during the test).

4.6.1.1 Resource requirements

First, the results for the use of resources will be presented. As mentioned in Section 3.6.2.1, Mini is equipped with 16 GB of RAM, and an Intel i5-3550 CPU with four cores running at 3.3 GHz, and uses Ubuntu 16.04 64 bits as its operating system. The Expression Manager is conformed by seven modules: Expression Scheduler, Expression Executor, Joint Player, TTS Player, Eyes Player, Touch Screen Player, and LED player. At any given time, there is one instance of each module, with the exception of the Joint Player, which requires individual instances for each joint (in Mini, there are five joints: head, neck, left and right arm, and base). The CPU and RAM requirements will be

presented individually for each module. The measurements provided for the Joint Player represent the resources used by a single instance of the player. Due to the fact that the process performed by every instance of the Joint Player is the same, regardless of the joint controlled, the total amount of resources used when all the joints are in motion can be computed as the product of the resources consumed by one instance of the player and the number of joints that the robot has.

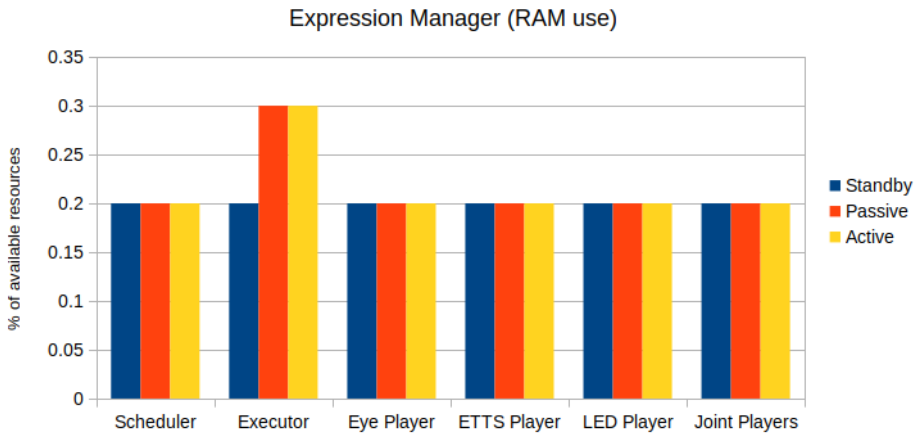


Figure 4.7: Percentage of RAM used for each module in the Expression Manager under the three conditions considered.

The three conditions under which the tests were conducted coincide with the ones used for evaluating the HRI Manager: (i) under the standby condition, the Expression Manager is deployed independently; (ii) under the passive condition, the software architecture is launched, and the robot is kept in a standby state; and (iii) under the active condition, the software architecture is launched and the robot is in operation. All the modules in the Expression Manager show a stable use of RAM, around 0.2-0.3% of the available memory. This is depicted in Figure 4.7. Regarding the CPU use, there is more variation from module to module, and also between conditions. Both the **Expression Scheduler** and the **Eye Player** showed a variation of CPU usage between 0.0 and 0.7% of the processing capacity provided by one of the CPU's cores under all three conditions. Similar values were measured for the **TTS Player** under conditions 1 and 2, while showing an increase under condition 3 to a maximum value of 1.3%, and for the **Touch Screen Player** under conductions 2 and 3, while under condition 1 the CPU use variated between 0.0 and 1.1%. The **Expression Executor** shows a significant increase on CPU use when the expressiveness architecture is being used. Under the condition 1, the Executor consumed between 0.0 and 0.7% of processing capacity. This percentage increased to a value between 0.7 and 2.0% under condition 2, and to a value between 2.7% and 18.4% under condition 3. The increase under the last condition could be tied

to the loading, configuration, and execution of FlexBE state machines. The **LED Player** showed a CPU use between 0.0 and 1.1% under condition 1, between 0.0 and 0.7% under condition 2, and between 0.0 and 1.2% under conduction 3. Finally, the **Joint Player** showed the highest CPU use, ranging from 1.9-2.0% and 3.3-3.4% under conditions 2 and 3. The fact that this player is the one that more resources uses can be attributed to the fact that is the only module that has to track the state of the action send to the interface, and also provides feedback at a 10 Hz rate. A summary of the CPU usage is shown in Figure 4.8.

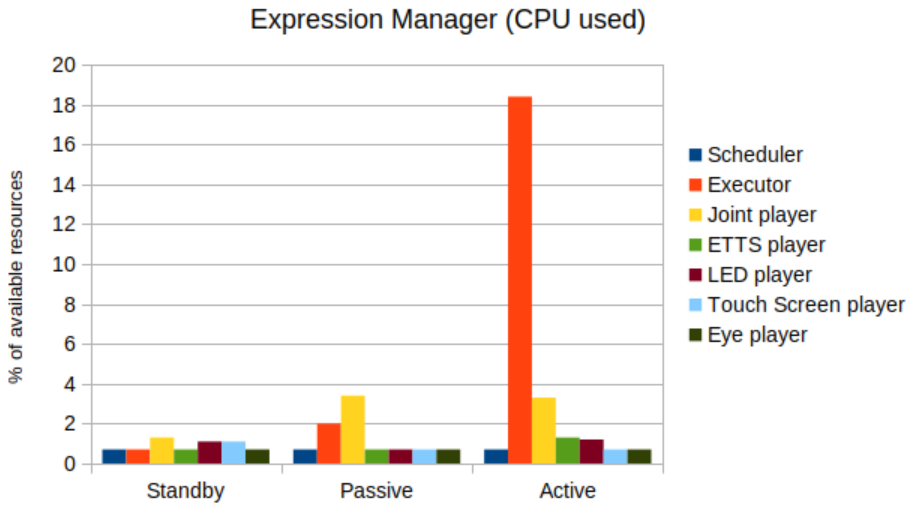


Figure 4.8: Peak use of CPU for each module in the Expression Manager under the three conditions considered. The use of CPU is computed as a percentage of the processing power of a single core.

Overall, considering the worst possible situation, the proposed expressiveness architecture requires 1.5% of available RAM, and 42.2% of one of the CPU's cores (assuming that the expression being performed requires the use of all interfaces at the same time, which is not very realistic). While the memory space used is acceptable, it would be beneficial to optimize the consumption of CPU. Because 70% of all the processing power consumed by the Expression Manager is used by the Expression Executor, this is the first module that should be improved (in particular, the process of loading and executing the state machines representing the robot's expressions).

4.6.1.2 Response time

Section 3.6.2.2 introduced the importance of endowing a robot with a system that allows it to respond in a short enough time to stimuli coming from the user or the environment. On one hand,

this manuscript already presented the *"two second" rule* as a criteria for the maximum delay between dialogue turns in a conversation. But for an expressiveness module this might be only part of its duties, as the robot might need to use expressions as a reaction to a perceived event, and this should be performed as soon as possible. According to some researchers [160], reaction time in humans tends to be around 200-250 milliseconds for simple reaction times (responding to stimuli appearing suddenly in the environment), and around 380 milliseconds for recognition reaction times (situation where the person has to discriminate between stimuli that should elicit a response, and those that should not). Based on these reports, three thresholds will be used in this section to measure the performance of the Expression Manager. On one hand, the combination of the HRI and Expression Managers response times should be around one second for it to be considered acceptable. On top of that, although the expressiveness architecture was designed mainly with conscious interactions in mind, it is interesting to evaluate if the performance of the system would allow to replicate the reaction times that can be observed in humans.

Similar to the strategy followed for measuring reaction times in the HRI Manager, loggers have been added to different points in the process of executing expressions. These loggers provided the timestamps that were used for computing the durations of the tasks performed by each individual module in the Expression Manager involved in the execution of the expression. These tasks are:

- **Schedule Action:** the time required for managing the execution requests that are sent to the Expression Scheduler. This task starts when the request is received in the callback, ends when the activation command is sent to the Expression Executor, and involves checking possible conflicts between the new expression and the active ones, and any action that has to be taken to correct these conflicts (storing the request in the priority queues, or stopping one or more of the active expressions and storing them in the queues).

- **Preparation:** time required for managing the tasks performed by the Expression Executor that are not handed by FlexBE. These tasks include extracting the activation command from the priority queue, checking that the expression is not already in use, checking if the activation request includes restrictions about the interfaces that can be used (for example, performing a gesture without using the voice), and loading the appropriate template from the gesture library.

- **FlexBE config:** time required for building and configuring the expression, using the methods provided by FlexBE. This task ends when the expression is ready to be executed.

- **Send action to player:** time required to execute the state machine that represents the expression and send the first action to the corresponding player.

- **Send action to interface:** time required to handle a goal received in the Interface Players. Is the duration between the moment in time when the goal is received in the player, and the moment when the message is sent to the corresponding output interface.
- **Clean Scheduler:** time that passes from the moment the expressiveness system receives confirmation that the last action in the expression has been completed, until the Expression Scheduler receives and handles the message that notifies the end of the expression.
- **Clean Executor:** time that passes from the moment the expressiveness system receives confirmation that the last action in the expression has been completed, until the Expression Executor receives and handles the message that notifies the end of the expression.

On top of these elements, the results also include an extra category called *Reaction time*. This is computed as the addition of the times required for completing the first five tasks, and represents the time that passes from the moment the expression request is received in the Scheduler until the first action in that expression is sent to the output interface. This will be the time that will be compared to the thresholds introduced above, as it represents the delay that will perceive the user whenever an expression has to be performed. Measurements were taken in six different trials. The first five involve sending requests to the Expression Manager to perform individual actions through interfaces managed by each of the five players. This is the most basic expression that can be performed, and represents the most favourable situation that the Expression Manager will face. It is also the most common in the current implementation of the software architecture, as the majority of interaction requests handled by the HRI Manager and sent to the Expression Manager involve using simple actions through a single interface, or a combination of a limited set of actions. In these five conditions, the expression used is the *customizable gesture*, presented in Section 4.5.4.1. In the last trial conducted, one of the predefined expressions was requested, in order to evaluate how the use of a complex expression that involves multiple actions synchronized among them affects the reaction time of the robot.

The results for the trials involving the execution of individual actions, which can be seen in Figure 4.9, show similar results for most of the tasks measured. In almost all cases, the time required for scheduling an action and for the different players to send commands is significantly shorter than the time required to complete the other tasks (around two or three orders of magnitude, depending on the task). The only exception is the Joint Player (in the trials involving this player, the action requested was to move the right arm up and down once), which requires around 0.1 seconds to send the first position in the trajectory to the motor. The reason behind this is that the Joint Player checks that the feedback provided by the motor is being received correctly before sending the

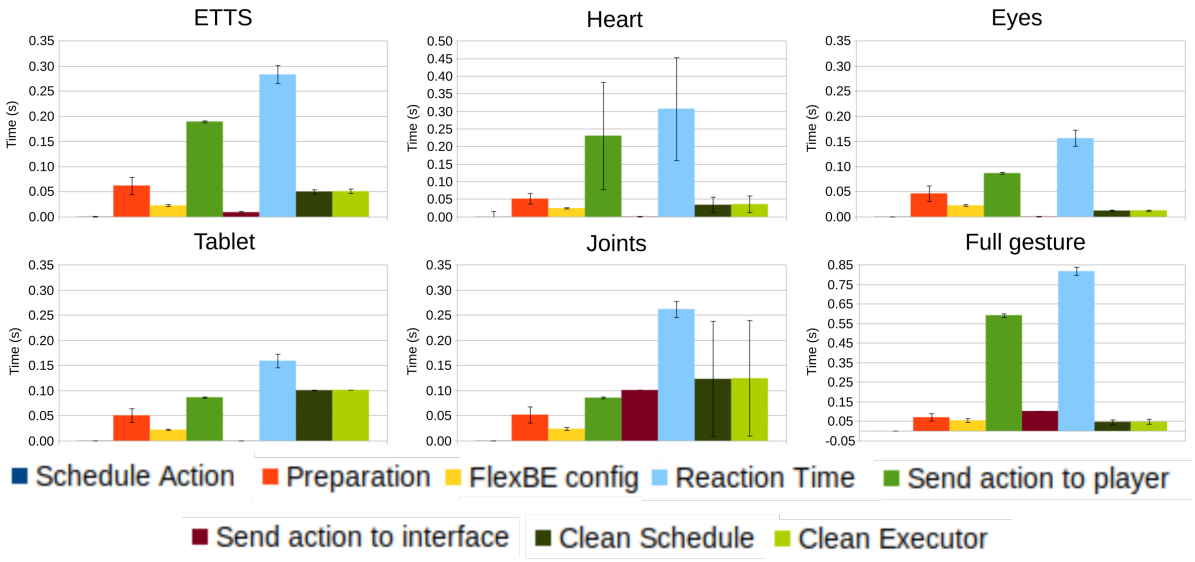


Figure 4.9: Duration of the different stages involved in executing an expression. The bars represent the average value, while the whiskers represent the standard deviation. Reaction time is the time since the expression manager receives the execution request until the first action is sent to the robot's interfaces.

command, which introduces a small delay. While the majority of the remaining tasks show similar durations between trials (something that could be expected, as most of these tasks are independent from the actual structure of the expression requested), the one that shows the highest disparity is the time required to initialize an expression and send the action to the corresponding player. The times measured oscillate between 0.085 seconds for motion actions and 0.23 seconds for LED configuration commands. This variation was not expected, as the expression used in all five trials is similar: a *customizable expression* conformed by an *action state* that sends the command to the player and a *control state* that checks that the player has completed its goal. Because the only difference is the *action state* used, variations should be attributed to the design of these states. Finally, it can be observed that the *Clean Scheduler/Executor* times for the expression that uses the touch screen and the one that uses the joints are significantly higher than for the other expressions. While for the expression using the tablet this could be explained due to delays introduced by the control loop that checks that the content sent to the tablet has been displayed for the specified amount of time, the reason for the increase observed for the joints has to do with the presence of one measurement that was significantly higher than the others, which leads to a high deviation.

In the last trial, the expression executed involved the motion of the right arm in a waving motion and also the motion of the neck from side to side. Each interface receives a sequence of commands that represent the different points in the trajectory. While current expressions can send multiple

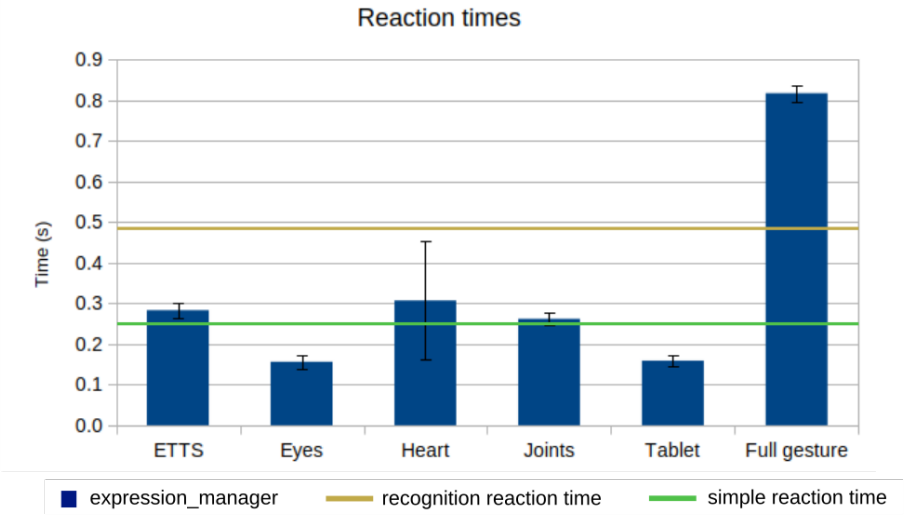


Figure 4.10: Time since the expression manager receives the execution request until the first action is sent to the robot's interfaces. The bars represent the average value, the whiskers represent the standard deviation, and the horizontal bars represent the thresholds for reaction times in humans.

positions to the Joint Player through a single *action state*, this expression was one of the earlier designs, and still uses individual *action states* for each position in the trajectory. Thus, it serves as an example of the system's reaction times when managing an expression that has been designed without having optimization in mind. Overall, the gesture used is composed of a sequence of ten *Concurrency* FlexBE containers, each containing two *action states* that are executed in parallel (one for the neck and another for the arm). These containers are separated by *wait states* that introduce delays between them, so the interfaces have enough time for completing the motions before receiving new commands. The results obtained are shown in the bottom right graph in Figure 4.9. While the times measured for the big majority of the tasks are similar to those obtained for the *customizable expression* that performs arm motions (which makes sense, as the player used is the same), there is a significant increase in the time required to start the execution of the expression. This can be attributed to the increase of the gesture's complexity and the extra amount of states that have to be initialized. This highlights the importance of optimizing the design of the expressions, as the addition of unnecessary states will hinder the performance of the Expression Manager. Figure 4.10 compares the reaction times measured in each of the six trials.

The reaction time that the user will be able to appreciate corresponds to the combination of the reaction times for the HRI Manager, the Expression Manager, and the robot's output modules, described in Chapter 2. This combination is the one that has to be compared with the thresholds

presented at the beginning of this section. In the results presented in Section 3.6.2.2, the worst case scenario observed for reaction times in the HRI Manager was around 0.3 seconds. This means that, in order for keeping the time required to perform an action below the 1 second threshold, the time required to send an action requested by the CA to the interface should be kept around 0.5 seconds (this is a conservative estimation that assumes that the action will be performed in about 0.2 seconds once it has been sent to the interface). For the trials where the *customizable expression* was used to convey individual actions, the worst case scenario observed occurred for the expression that sent commands to the LED heart (although the value has a remarkable high deviation, due to the presence of two measurements that were significantly higher than the others). In any case, the expression was performed in 0.45 seconds or less in every attempt, which is below the threshold defined for interactions managed by the HRI Manager. Regarding the reactive expressions, the time required to perform actions involving the tablet or the eyes of the robot is close to the simple reaction times measured in humans, and always below the reaction times for recognition tasks. For actions involving the TTS or the joints, the time was slightly higher than the simple reaction time threshold, but still below the recognition reaction time. Finally, for the heart, the time was higher than both thresholds, although this is due to the presence of two measurements that distort the results obtained. Finally, for the last trial (the execution of a complex expression), the reaction time measured is around 0.82 seconds, well above the 0.5 second threshold defined. This means that, under the worst conditions encountered during the experiments, the combined reaction time for the HRI and Expression Manager would be around 1.16 seconds, above the one second threshold defined in Section 3.6.2.2, but still below the limit defined by the *"two second" rule*. Also, this indicates that complex expressions cannot be used to show an immediate reaction to a stimulus suddenly appearing in the environment. Figure 4.11 shows a summary of the reaction times for the HRI Architecture as a whole (the combination of the HRI Manager and the Expression Manager), along with the threshold defined in this dissertation for a natural interaction and the threshold defined by the *"two second" rule*.

Based on the results obtained from the objective evaluation, it can be concluded that the performance of the Expression Manager when handling gestures involving individual actions (which, as stated before, is the most common situation for the robotic platforms used in this thesis) is enough to guarantee an interaction that satisfies the temporal constraints involved in human-robot communication, and close enough to the reaction times that can be observed in humans for reactive expressions. Regarding the use of complex expressions, the results suggest that they can be used in conscious interactions between the robot and the human while maintaining an acceptable performance, but should not be used to react to unexpected events that require an immediate

response. The results also suggest that, in order to optimize the performance of the proposed expressiveness architecture, the bottleneck that should be reviewed and improved is the process for initializing the state machines that are used to represent the expressions.

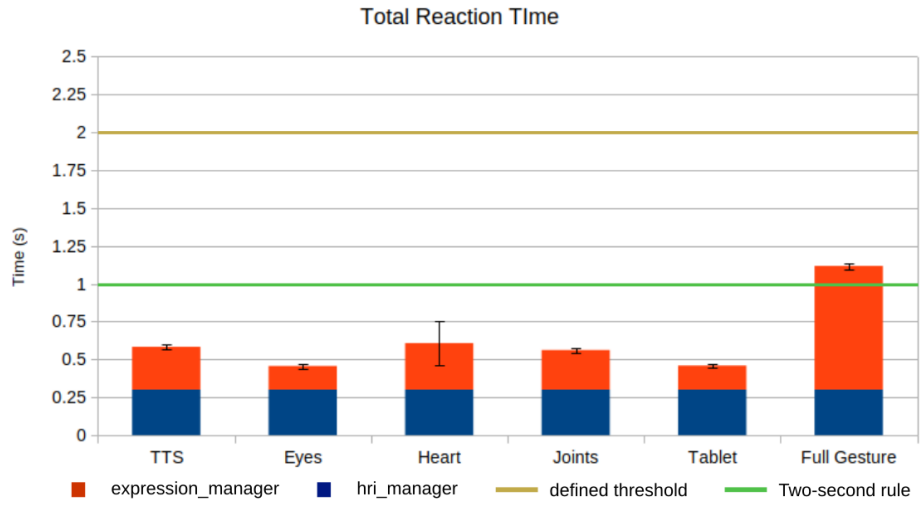


Figure 4.11: Reaction time for the combination of the HRI Manager and the Expression Manager. The bars represent the average value, the whiskers represent the standard deviation, and the horizontal bars represent the two thresholds defined for interactions between the robot and the human.

4.6.2 Subjective evaluation: effect of the parameter-based modulation

The first subjective evaluation was conducted to evaluate two of the modulation mechanisms that have been described in Section 4.5.4. The first method allows developers to replace actions from a predefined expression with new values, in order to adapt that gesture to the context of the interaction. The second mechanism can be used to modify the appearance of an expression with the variation of two modulation parameters: speed and amplitude. The goal of the experiment is to evaluate if an appropriate modification of the robot's expressiveness using these two methods can improve the overall perception that users have of a social robot that is endowed with a limited amount of different expressions.

Due to the healthcare concerns caused by the COVID-19 pandemic, this study has been conducted using video-based evaluations. This simplifies the process of adding participants to the experiment, providing a larger body of results, and has also the advantage of ensuring the repeatability of the experiment between participants (in-person evaluations can introduce small variations in the

experiment conditions due to real world factors that could affect the perception of the participants). On the other hand, video-based experiments can fail to capture all the details of the experimental conditions, providing a less realistic experience, and thus leading to less accurate results.

4.6.2.1 Experiment definition

Figure 4.12: Mini playing the landmarks game with a user.

The interaction featured in the videos used in the experiment involved one of the cognitive stimulation exercises introduced in the case of use presented in the subjective evaluation of the HRI Manager. In this exercise, the robot presents a series of pictures of famous landmarks from all around the world, and the user has to guess in which city they are located. Figure 4.12 shows the setup during the interaction. At the beginning of the video, the robot is seen in a standby state, waiting for the user to start the interaction, while the user is sitting in front of the robot, off-screen. The user starts the interaction by waking the robot, rubbing its shoulder. Mini wakes up (performing the *wake up* expression), introduces itself, notifies the user that they are about to play a game, and finally explains the rules of the landmarks exercise. Next, the robot moves on to the first question of the exercise. In this case, Mini showed in the tablet an image of the Eiffel Tower, and offered the user the following options: Rome, Paris, and Madrid. The user was fairly convinced about the correct

option, and answered *"I think it's in Paris"*. This led the robot to congratulate the user for providing the right answer. This was done with the *congratulate* expression, which has the robot raising the arms, shaking them, and saying *"Very good, that is the correct answer"*. Then, the robot continues with the second question of the exercise, saying *"Lets move on to the next question"*. Here, the image shown is from the Colosseum, located in the Italian city of Rome. The options presented to the user are: London, Beijing, and Rome. Again, the user comes on with the correct option, which in turn makes Mini congratulate the user with the same *congratulate* expression, and moves to the next question, repeating the same sentence uttered between questions 1 and 2. In the third question, Mini asks the user about the location of the Leaning Tower. In this case, the answer provided by the user was wrong, and the robot used the *regret* expression to provide feedback. This gesture makes the robot shake its head and say *"No, that is not the correct answer"*. Once again, Mini repeats the *"Lets move on to the next question"* sentence during the transition between the third and fourth question. The last picture shown to the user is from the Roman aqueduct that is located in the Spanish town of Segovia. The user is not sure about this answer either, and again selects the wrong option. Mini repeats the *regret* expression, thanks the user for completing the exercise, and greets him goodbye.

4.6.2.2 Conditions

Two different conditions have been defined. Under the *neutral* condition, the interaction shown in the video occurs as described in the previous section. Mini uses the predefined versions of the *congratulate* and *regret* expressions to provide feedback to the user, repeats always the same exact sentence in the transition between questions, with the same inflection, and performs every single action using the *normal* setup for both the amplitude and the speed modulation parameters. This means that the features of each communicative channel described in Table 4.3 will be kept unchanged during the whole video.

The *expressive* condition, on the other hand, makes use of the modulation mechanisms to improve the expressiveness of the robot. The method for replacing actions in the predefined expressions is used to adapt the gesture used for transitioning between questions to the context of the interaction. While the transition between the first two questions is left unchanged, after the second question the utterance is replaced by a new one that acknowledges that the user has provided two consecutive correct answers. The new transition between the third and fourth question Mini tries to comfort the user for giving a wrong answer, and encourages him to do better in the last question. Finally, the last transition is modified to mention that the user has given two consecutive wrong answers. The actions performed by both the questions and the expressions used to provide feedback

to the user after each answer are left unaltered. This does not mean that they will not be modulated, only that the expressions are built with the predefined actions.

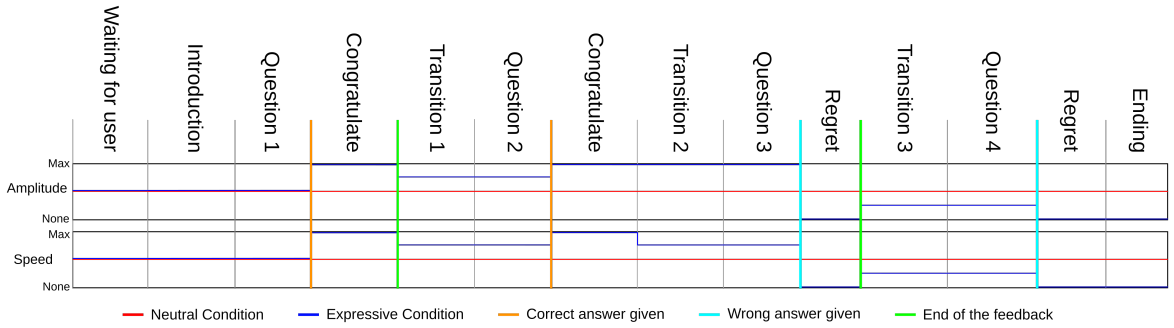


Figure 4.13: Evolution of the modulation parameters (speed and amplitude) during the entire experiment for both conditions. The red line represents the variation of the parameters under the *neutral* condition, and the blue line represents the variation of the parameters under the *expressive* condition.

The mechanism that allows to modify the appearance of the gesture based on the values assigned to the control parameters is used to alter the actions performed by Mini depending on how the user is doing at the game. The feedback gestures are considered to be expressions that should convey the extreme values of intensity, as the user giving the correct or wrong answer is considered to be the stimuli that will be used to change the robot's expressiveness. Table 4.4 shows the effects that these stimuli have over the modulation parameters. In order to convey the idea that Mini is happy whenever the user gives a correct answer, both the speed and amplitude will be set to *high increase* when performing the *congratulate* expression. On the other hand, the *regret* expression will always be performed with a *high decrease*, to represent that Mini is sad because the user gave a wrong answer. The values of speed and amplitude used for the rest of the expressions performed during the interaction will be continuously updated to represent the state of the game. Both parameters will be set to *normal* at the beginning of the interaction. After the user gives the first correct answer, and the robot performs the *congratulate* gesture, speed and amplitude are set to *medium increase*. This will affect the transition between questions 1 and 2, and also the expression used to ask the second question. The idea is that the happiness that the robot conveys with the *congratulate* gesture after the correct answer diminishes after the expression is completed, but is still present in the actions of the robot. After Mini performs the second *congratulate* expression, speed is kept at *medium increase*, but amplitude is raised to *high increase*. This tries to convey the idea that the robot is happier because the correct answer streak continues. While right answers increase the appearance of happiness, wrong answers produce the opposite effect, this is, they make the robot look sadder. After

the first wrong answer was provided in question 3, the speed and amplitude parameter are both set to *medium decrease*. After the second wrong answer was given for the last question in the exercise, both parameters are lowered to *high decrease* for the remainder of the interaction. The experimental conditions are *between-subjects*, which means that each participant will only evaluate one of the conditions, selected randomly.

Stimulus	Effect
Correct answer	Increase the speed and amplitude of the expressiveness. The effect is cumulative.
Wrong answer	Decrease the speed and amplitude of the expressiveness. The effect is cumulative.

Table 4.4: Stimuli considered in the experiment for altering the amplitude and speed of the robot's expressiveness.

4.6.2.3 Questionnaire

The evaluation of the conditions presented above has been done using online questionnaires. The first section of the questionnaire includes a series of fields for retrieving demographic information: age, gender, education level, their level of familiarity with technology in general, and with robots in particular, and their level of interest in interacting with a robot and in owning one. A control question was included to detect participants that already knew the robot, as their responses could be biased. Once the first section of the questionnaire is completed, the participants are presented with the video corresponding to one of the conditions, and asked to watch the entire interaction before advancing to the next question.

The evaluation of how the participants perceived the robot in the videos was done using the Robotic Social Attributes Scale (ROSAS) [161]. In this questionnaire, the evaluation of the robot is performed according to 18 items regarding different social attributes that the robot might present. For each item, the participant is presented with two antonyms placed at opposite ends of a scale (e.g., *artificial-likelife*). The 18 items are then used to compute three scale dimensions: warmth, competence, and discomfort. In the questionnaire used in this evaluation, the 18 pairs were extracted from the English version of the RoSAS, and then translated into Spanish. On top of this, an extra control question was added to guarantee that participants watched the entire video. In this question, the participant has to indicate the amount of different robots that appear during the whole

interaction. The last section in the questionnaire included a text box where participants were allowed to provide comments and suggestions oriented to improve the robot. Participants were randomly distributed, and the number of participants in each condition was balanced.

4.6.2.4 Hypotheses considered

Based on the sub-scales of the RoSAS questionnaire and the conditions designed, the results obtained will be used to evaluate the following hypotheses:

- **H1:** The modulation of the robot's expressiveness will affect the perception that the participants have of the robot.
- **H2:** The participants' perception of *warmth* will be higher under the *expressive* condition than under the *neutral* condition.
- **H3:** The participants' perception of *discomfort* will be higher under the *neutral* condition than under the *expressive* condition.
- **H4:** The participants' perception of *competence* will not show significant differences between both conditions.

4.6.2.5 Participants

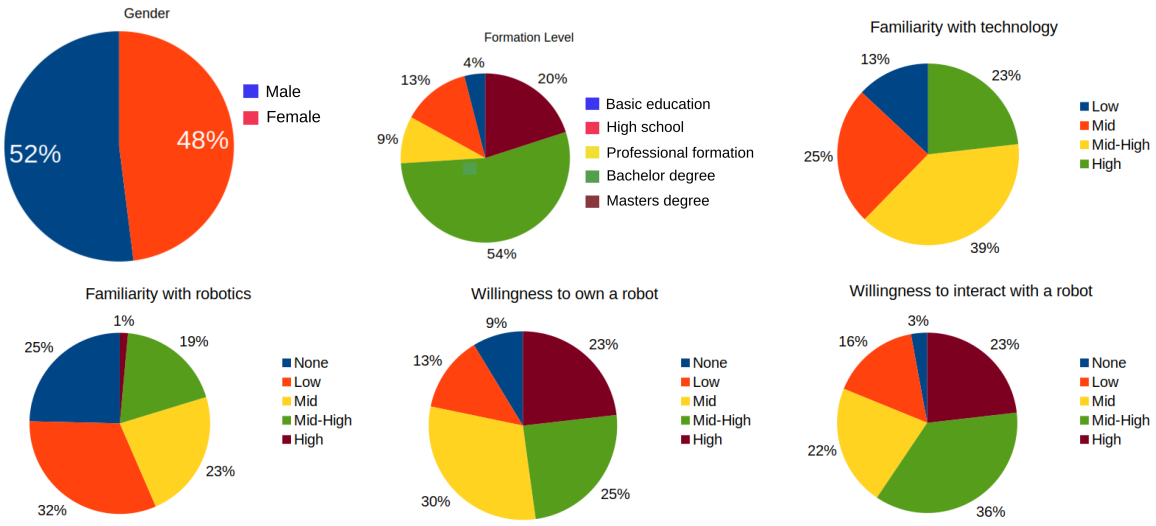


Figure 4.14: Demographic analysis of the participants in the experiment.

Participants were contacted through email and messaging applications for this experiment. In total, 69 persons completed the questionnaire, split evenly between both conditions (34 for the *neutral* condition, and 35 for the *expressive* condition). Out of these 69, 36 were male and 33 female. 4% of the participants have only basic education, 13% were high school graduates, 9% had professional formation, 54% were college graduates, and 20% were master graduates. A large majority of participants had either a medium or medium-high familiarity with technology in general (64%), while a 23% had a high level of familiarity. However, close to an 80% of participants reported having a medium or lower familiarity with robotics. Finally, 81% of participants reported medium or higher interest in interacting with a robot, and almost the same amount of participants reported medium or higher interest in owning a robot. A summary of these results is shown in Figure 4.14

4.6.2.6 Results obtained

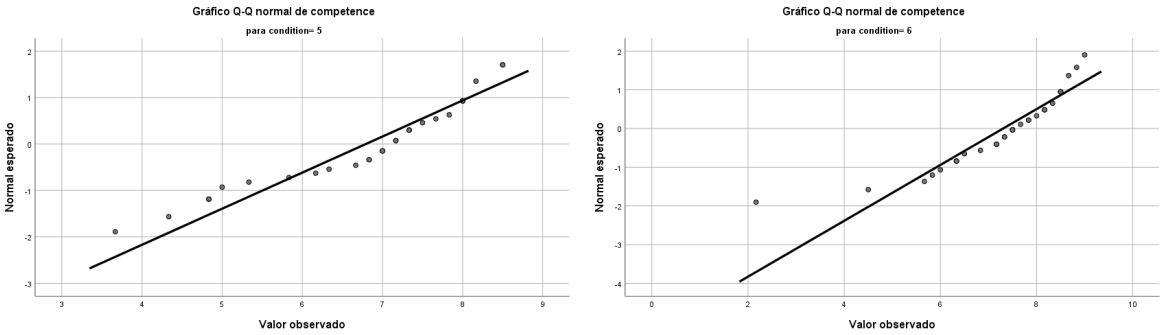


Figure 4.15: Q-Q graphics for the *competence* dimension under the *neutral* (left) and *expressive* (right) conditions. The line represents a perfect normal distribution.

Before starting the analysis, data was pre-processed to identify duplicated cases. Two of the responses were marked as duplicated and removed. Next, based on the 18 item pairs evaluated in the questionnaire, the three scale dimensions of the RoSAS, *warmth*, *competennce*, and *discomfort*, were computed. First, normality tests were performed for the three dimensions under both conditions. While the Saphiro-Wilk test showed that the *warmth* ratings did not deviate significantly from normal ($W_{neutral}(33) = 0.948, p = 0.118$ and $W_{expressive}(34) = 0.951, p = 0.132$), the same cannot be said for the other two dimensions. In order to correct this, a squared root transformation was applied to both the *competence* and *discomfort* dimensions. The Saphiro-Wilk test was repeated for the transformed variables (*sqrt_competence*, and *sqrt_discomfort*). For *sqrt_discomfort* now follows a normal distribution ($W_{neutral}(33) = 0.954, p = 0.178$ and $W_{expressive}(34) = 0.944, p = 0.08$). Although the results obtained for *competence* did not fall under the normality assumption, looking at

its normal Q-Q plots for both conditions (see Figure 4.15), it can be observed that the data does not present a significant deviation from the diagonal line that represents a perfect normal distribution, and thus it will be considered from here on that the normality assumption is met for all three dependent variables (*warmth*, *competence*, and *sqrt_discomfort*).

Figure 4.16: Ratings for the three sub-scales in the RoSAS questionnaire (warmth, competence, discomfort) for the expressive and neutral conditions. The bars represent the average value and the whiskers represent the 95% confidence intervals. The asterisk indicates the significant differences between conditions.

The first evaluation of the results conducted was a review of descriptive statistics for the RoSAS dimensions under both conditions. The result is shown in Figure 4.16. The addition of the modulation causes an increase in both warmth and competence. The average rating for discomfort is also higher under the *expressive* condition, although the difference between conditions is smaller for this dimension than for the other two.

Independent Samples T-Tests were conducted for the *warmth*, *competence*, and *sqrt_discomfort* variables in order to find if there are any significant differences between conditions. Levene tests showed that the variances for all three dependent variables are equal. The results for the T-Test show that there is a significant difference between the neutral and expressive conditions for warmth ($t(65) = -2.173, p = 0.033$), while competence ($t(65) = -1.563, p = 0.123$) and sqrt_discomfort ($t(65) = -0.327, p = 0.745$) did not show significant differences. These results indicate that the modulation of the robot's actions can lead to a higher perception of warmth in the robot.

Next, the responses to the questionnaire were divided based on the answers to the questions included in the demographic information section, in order to evaluate the effect that each of those characteristics had on the perception of the robot. An analysis of covariance (ANCOVA) was conducted to compare the ratings of all three dimensions (*warmth*, *competence*, *sqrt_discomfort*) using the variables *familiarity with technology*, *familiarity with robotics*, *willingness to interact with a robot*, and *willingness to own a robot* as co-variables. The following results were extracted from this evaluation:

1. There are significant differences in the rating of *warmth* when the familiarity with technology is also controlled ($F = 4.496, p = 0.038, \eta^2 = 0.066$).

2. There are significant differences in the rating of *warmth* ($F = 4.555, p = 0.037, \eta^2 = 0.066$) and marginal differences in the ratings of *competence* ($F = 3.845, p = 0.067, \eta^2 = 0.051$) when the familiarity with robotics is also controlled.

3. There are significant differences in the rating of *warmth* when the willingness to interact with a robot is also controlled ($F = 4.402, p = 0.04, \eta^2 = 0.064$).

4. There are significant differences in the rating of *warmth* when the willingness to own a robot is also controlled ($F = 4.687, p = 0.034, \eta^2 = 0.068$).

For each of the four co-variables that showed differences (almost all of them significant), extra tests were conducted to obtain more information. All four variables can take five possible discrete values (None, Low, Medium, Mid-high, and High). The results obtained were divided in two subsets for each co-variable, *High* and *Low*. The division was made trying to ensure that the sizes of both subsets were as similar as possible. For the *familiarity with technology* variable, it was considered that participants with low familiarity were those that answered *None*, *Low*, or *Medium*, while participants with high familiarity were those that answered *Mid-high*, or *High*. For the familiarity with robotics, the participants that answered *Medium* were included in the group of high familiarity. The splits based on the willingness to interact or own a robot follow the same distribution that the one for the familiarity with technology. A summary of how the data was divided into the *Low* and *High* category based on the responses given for each co-variable can be seen in Table 4.5.

This resulted in 8 different subsets of data. For each of them, Independent Sample T-Tests were conducted, with Levene's tests to ensure the homogeneity of variances. In all cases, the assumption of homogeneity is met. For the variable *warmth*, the following results were obtained:

	Low	High
Familiarity with technology	None, Low, Medium	Mid-High, High
Familiarity with robotics	None, Low	Medium, Mid-High, High
Willingness to interact with a robot	None, Low, Medium	Mid-High, High
Willingness to own a robot	None, Low, Medium	Mid-High, High

Table 4.5: Subsets of data created based on the values of the four co-variables included in the demographic section of the questionnaire.

1. For the participants that reported a low familiarity with robotics, there is a significant difference on the rating of warmth between the *neutral* and *expressive* conditions ($t(37) = -2.835, p = 0.007$).

2. For the participants that reported a low familiarity with technology, there is a significant difference on the rating of warmth between the *neutral* and *expressive* conditions ($t(24) = -2.917, p = 0.008$).

3. For the participants that reported a low interest on interacting with a robot, there is a significant difference on the rating of warmth between the *neutral* and *expressive* conditions ($t(26) = -2.064, p = 0.049$).

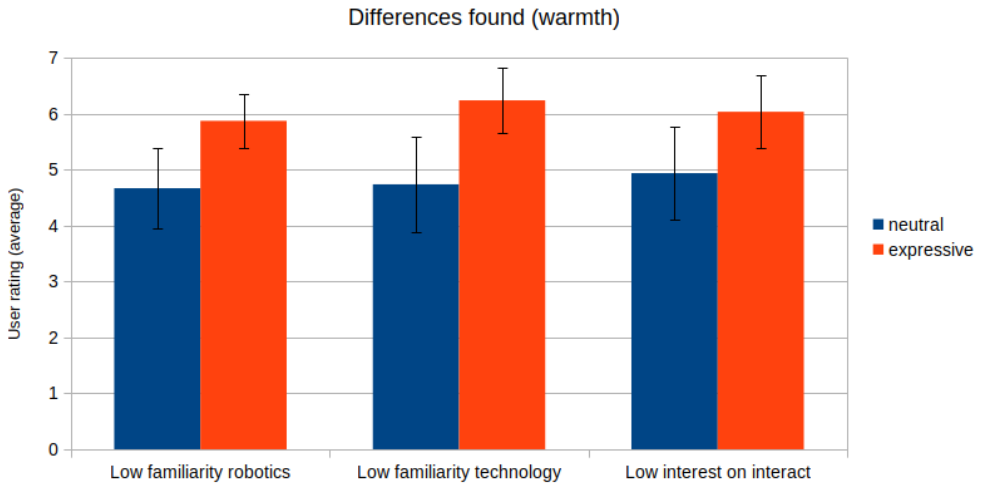


Figure 4.17: Significant differences found through the Independent Sample T-Tests for the *warmth* dimension. The bars represent the average value and the whiskers represent the 95% confidence intervals. The asterisk indicates the significant differences between conditions.

Figure 4.17 summarizes the significant differences found for the *warmth* dimension. On the other hand, for the variable *competence*, the following results were obtained:

1. For the participants that reported a high familiarity with technology, there is a marginal difference on the rating of competence between the *neutral* and *expressive* conditions ($t(39) = -1.895, p = 0.066$).

2. For the participants that reported a high interest on owning a robot, there is a significant difference on the rating of competence between the *neutral* and *expressive* conditions ($t(29) = -3.122, p = 0.004$).

Figures 4.18 shows a summary of all the differences found for the *competence* dimension.

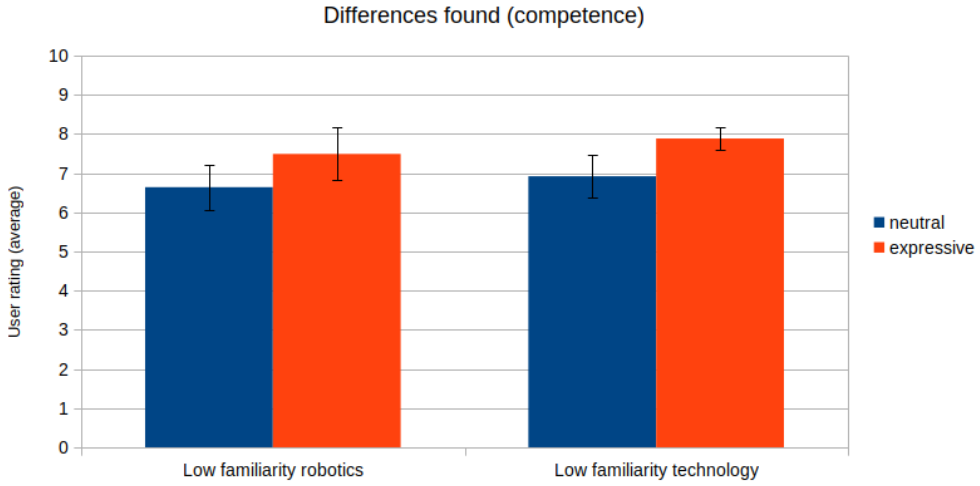


Figure 4.18: Differences found through the Independent Sample T-Tests for the *competence* dimension. The bars represent the average value and the whiskers represent the 95% confidence intervals. The asterisk indicates the significant differences between conditions.

4.6.2.7 Discussion

Observing the overall ratings obtained for all three variables, participants did find the *warmth* of the robot to be significantly higher under the *expressive condition* than under the *neutral* condition, while no significant differences were observed for the other variables. This validates hypothesis **H1**, as the modulation did, in fact, provoke a change in how participants perceived the robot. The fact that the main effect of modulating the robot's expressiveness to adapt it to the context of the interaction is a higher rating in *warmth* and not in *competence* is not surprising, and validates hypotheses **H2** and

H4. Under both conditions, the robot is able to conduct the interaction with the user correctly, reacting to the answers of the user, and showing knowledge about the task in hand. Mini *knew* where each monument was located and was able to correctly assess if the answers given by the user were correct or not. Thus, it is reasonable to assume that the items that compose the *competence* scale of the RoSAS questionnaire (capable, responsive, interactive, reliable, competent, and knowledgeable) would not be highly affected by a change in how the robot expresses itself. Regarding the rating for *discomfort*, although it was expected that it would decrease with the addition of the modulation, the lack of significant variation could be due to the role that other factors play (for example, the external aspect of the robot, or the design of the interaction, among others). Another possible reason behind it might be an excessive modulation. An intense variation in how the robot performs actions in a short period of time (like in the interaction conducted during the study) could make the robot more expressive, but also make it look less natural if it is taken too far. Further tests would be required to obtain a better understanding of this. In any case, the lack of effect that the modulation has over the rating of *discomfort* translates in hypothesis **H3** not being validated.

Focusing now on the evaluation conducted while controlling the different co-variables, several conclusions can be extracted. First, it seems that participants less used to technology and robots seem to perceive the effect of modulation clearer, and find the robot to have more warmth when its actions are being adapted to the context of the interaction. A possible explanation for this would be that a lack of familiarity leads to not having a frame of reference for comparing the performance of the robot. Thus, participants that have had less contact with robotics (or technology in general) will not be able to compare Mini's expressiveness to that displayed by other robots. Second, participants that show interest in owning a robot or have a high familiarity with technology seem to believe the robot to be more competent when it is able to adapt the expressiveness to the circumstances of the interaction. Following the explanation given for the first conclusion presented, this could mean that those participants with more knowledge (or interest to know) about the robot's capabilities are more susceptible of perceiving the variation that the modulation of Mini's expressiveness introduces. Finally, the fact that participants that are less willing to interact with a robot are able to appreciate a difference between both conditions could be caused because this lack of interest makes them less inclined to evaluate positively the robot, and thus there is more range for improvement. In any case, the results obtained seem to indicate that there is a benefit to endowing social robots with the ability to adapt their behaviour to different situations, and that in Mini this can be achieved with the two modulation methods used in this study (replacing actions inside the expressions in runtime and using the *speed* and *amplitude* parameters to modify the appearance of the expressions).

4.6.3 Subjective evaluation: effect of the profile-based modulation

The goal of the second subjective evaluation conducted is to test the effectiveness of the third modulation approach. In this case, the Expression Manager will use modulation profiles to adapt the expressiveness of the robot to its internal state. In this case, the experiment focuses on the expression of affect states, both mood and emotion, as these are one of the most prominent examples of internal states that play a big part in shaping the actions of a social robot. Like in the previous study, a video-based evaluation was chosen due to the concerns raised by the COVID-19 pandemic.

4.6.3.1 Affect state expression

Mini has been equipped with an affect generator that governs the emotional state of the robot based on stimuli collected from the environment. The stimuli are classified inside an activation system (for example, cognitive processes, or sensorimotor information). The value of these stimuli will affect the valence and arousal of the robot, which in turn will determine the emotion/s active and the robot's mood. Under this model, the intensity of the affect state will increase in presence of a stimulus, and then decay at a certain rate when the elicitor disappears. Based on the levels of arousal and valence, the mood of the robot can be either *anxious*, *happy*, *bored*, or *relaxed*, while the emotions considered are *happiness*, *sadness*, *anger*, and *surprise*. There is also a *neutral* state that represents the idle state of the system, when no mood or emotion is active. This approach to affect state generation considers moods to be discrete, while emotions are continuous, and have associated an intensity level that ranges from 0.0 to 100.0.

In order to improve the integration of both affect states in the robot, the Players include a method for fusing the effects that moods and emotions have over expressiveness. If the affect generator notifies a change in the robot's mood, the Players load the configuration related to the new mood for each interface. In this approach, the robot will always be conveying a mood. When an emotion is triggered with the highest intensity, the expressiveness will be modulated to represent exclusively that emotion. Then, as the intensity of the emotion decays, the expressiveness will be continuously adapted so it shifts from conveying only the emotion to conveying only the mood that the robot is feeling, when the emotion disappears. In between, the modulation of the expressiveness will be the result of the combination of the mood and emotion effects, with the importance of the latter being proportional to the emotion's intensity. During the integration of the proposed approach, it was observed that there was a significant period of time between the moment when the effect of the emotions' intensity stopped being perceivable and the moment when the intensity actually reached

0.0. Thus, it was determined that if the intensity drops below an empirically defined threshold (5.0 in this case), the modulation will immediately switch to conveying only the robot's mood.

Also, when a new update on the robot's state is received in the emotion display module described in Section 4.5.5, the module starts by checking if there is a dominant emotion, and retrieves its intensity if it exists. If the intensity is above a threshold (the value used in this evaluation was 80.0), the appropriate expression is retrieved, and an activation request is sent to the Expression Manager. The reason behind the implementation of the threshold is the idea that a explicit emotional response to an stimulus should only be triggered by a highly intense emotional episode. These emotional expressions have been carefully designed to react to the specific stimuli that will appear during the experiment (the user gives a correct/wrong answer to a question, the user hits the robot, and the user caresses the robot). Each emotional episode can only trigger the execution of a single emotional expression.

4.6.3.2 Design of the affect-based modulation

As presented in Section 4.3.4, the relationship between affect states and expressiveness is a research topic that has attracted a significant amount of attention. This section introduces the knowledge that was used to define the specific effect that each emotion and mood has over the robot's expressiveness. According to the discussion presented in [151], Ekman proposed that there are six or seven basic emotions (the consideration of contempt as a basic emotion was not as clear as the other six: happiness, sadness, fear, disgust, anger, and surprise) that have associated prototypical expressions. Socially acquired display rules can affect the relationship between an emotional state and its prototypical expression. There is evidence that supports that there is an innate component of emotional facial expressions, although the connection between expressions and states is not completely understood. Another conclusion that can be extracted from [151] is that basic emotions tend to be recognized universally, although it can be observed that several emotions show variations on how they are displayed in different group. In any case. these differences are not big enough to hinder recognition.

Regarding the different communication interfaces that Mini can use, there have been multiple studies that evaluate the particular effect for each emotion. Regarding facial expressions, the pioneer is considered to be Charles Darwin. In [139], he stated that happiness is associated with the act of smiling, where the cheeks are raised, pulling the corners of the mouth up. Displaying sadness involve raising the inside corners of the eyebrows. An angry expression would be recognized by the eyes being wide open, the nostrils being raised, and a furrowed brow. Finally, an expression of surprise presents

open eyes and mouth, and raised eyebrows. Another feature connected with an expression of anger, as presented in [152], is the red margin of the lips becoming tighter and narrower. In that same work, Ekman argues that disgust, another of the so called *basic* emotions, can be conveyed through the raising of the nares, the pull up of the infra-orbital triangle, and the wrinkling of the nose's sides.

When considering the effect that affect states have over the speech, two components can be considered: content and prosody. Because the former is fixed by the robot's applications, the expression of affect in Mini will be achieved through the modulation of the latter. Emotions ligated to the arousal of sympathetic nervous system cause an increase on speech rate, voice volume, and also high frequency energy. If the parasympathetic system is the one aroused instead, which can happen when feeling sad or bored, then the opposite happens: low voice, pitch and high frequency energy, and slower rate [162]. These findings are confirmed by other works [163, 164] that stablish how *surprise* and *anger* lead to an increased pitch, energy, and speech rate, *joy* also causes an increase in the first two features, while its connection with the rate of articulation is not always present. Finally, *sadness* leads to a decrease of all three characteristics.

The work of Darwin [139] also studied the effect that emotions have over body motions. A passive attitude characterized by a small amount of motion or a posture with the head hanging on the chest are indicators of sadness, while anger would produce whole body trembling, clenched fists, squared shoulders, and frantic movements. Regardless, not many researchers focused in this particular channel of communication. In the survey conducted by Kleinsmith and Bianchi-Berthouze [155], some authors report that anger is conveyed by bending the head forward and also bending the elbows, while other work reported that the head is bent back when in anger. Happiness is connected to the head being bent back and the arms being raised, according to several authors, while others mention an upright position of the head and the arms being straight. However, although some authors report similar body features associated to a given emotion, not all reports coincide.

The final modality considered is the use of colors to convey affect. The results of the study conducted by da Pos and Green-Armytage [156], which were collected using participants from Europe and Australia, indicate that there is not a one-to-one relationship between colours and emotions. Anger and happiness generated a greater consensus than the rest of the expressions. For anger, participants selected mainly reddish colours, while happiness was associated with red and yellow tones. Surprise showed a similar choice of colours, although the reds selected were closer to magenta, instead of orange. A majority of participants selected blue tones for sadness. Finally, the colour selections for disgust and fear are more distributed. Other works evaluated the emotional

response that each colour evokes on a person. In the study conducted by Valdez and Mehrabian [157], the arousal dimension showed a direct relationship with colour saturation and an inverse relation with brightness. On the other hand, pleasure was a joint positive function of both brightness and saturation. Regarding hue, a relation was found between this colour feature and pleasure, while the relation with arousal was weaker. Finally, Sutton and Altarriba [158] conducted an experiment where participants were given 160 emotional terms, and were asked to select the colour they would associate said terms with. The results of the study showed a connection between the colour red and negative emotions, while yellow and white were connected to positive emotions.

Both the mood and emotion profiles that will be used to modulate the appearance of Mini's expressions have been designed to reflect the features presented above, to the extent that the robot is capable. For example, because Mini lacks a mouth, the expression of affect states through facial expressions had to be reworked to use only the eyes (only feature that can be controlled in Mini). An evaluation was conducted to ensure that the modulation designed transmits affect states correctly, The results of this evaluation will be presented in Section 4.6.3.7.

4.6.3.3 Experiment definition

In the interaction used to evaluate the robot in this study, the robot plays a quiz game with the user. In this game, the user can select a category, and the robot starts asking questions about it, giving the user four options to choose from. In the interaction depicted in the videos, the category selected is History. The length of the game was set to four questions, as this was enough to showcase all the possible affect states without overextending the interaction, which could be tiring for the participants. Figure 4.19 shows Mini during the game.

At the beginning of the video, Mini can be seen in an idle state that simulates being asleep, while a button is shown in the tablet with the text *"Press to start"*. After the user presses the button, Mini executes the *Wake up* gesture, introduces itself, and then starts explaining the rules of the game. Once the instructions have been explained, the robot asks the user to choose a category for the game. The user answers: *"I like history"*. Then, the robot confirms the user's selection, and starts with the first question: *"In what year did the attack on Pearl Harbour took place?"*. The user selects the option "1941" in the touch screen. This being the correct answer triggers the activation of the "Happiness" emotion. Mini congratulates the user, and then gives a small explanation about the attack on Pearl Harbour. Once this explanation has been completed, the robot asks the user about their opinion on the topic of the question, and the user answers that he likes questions about History. Mini shares

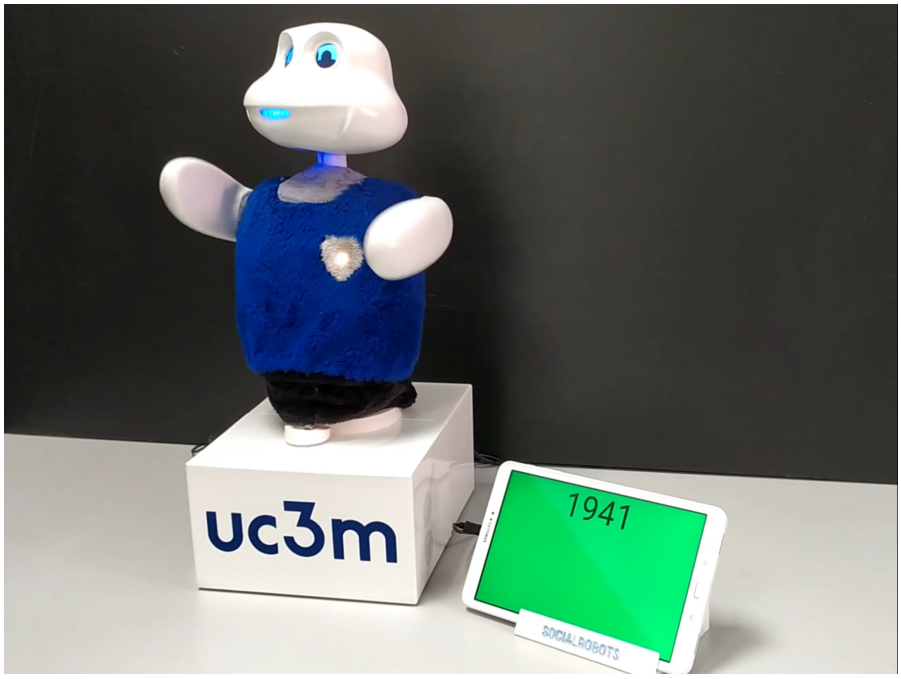


Figure 4.19: Mini playing the quiz game with an user.

with the user that it also likes learning about History, closing this small dialogue, and then moves on to the next question.

Question two was about which was the oldest city in South America. Although the correct answer was Caral, the user did not know this, and he answered "La Paz". This triggered the "Sadness" emotion, and made the robot perform the *Regret* gesture. After informing the user about the correct answer, Mini gave a brief explanation about this answer, and then asked the user a personal question (If he had visited Perú, the country where Caral is located). The user answered the question and the robot gave a small feedback (because the user said he had never visited Perú, the robot recommended him travelling there). Then the game moved on to the third question. This time, the user had to answer who the husband of Cleopatra was. Again, the user thought the right answer was Julius Caesar, when in reality all the answers were correct. This triggered the "Sadness" emotion, but this time, the user felt like he had been cheated, so he hit Mini, which triggered the "Anger" emotion. The robot admonished the user, and then continued with the explanation of the answer. After a small dialogue between robot and user with another personal question, the game moved on to the last question.

The last question was *"What age started with the Renaissance?"*. Now, the user was able to give the correct answer (the Modern Age"). Mini congratulated the user, showing happiness,

and in turn, the user celebrated with the robot, caressing its belly. This stimulus triggered the "Surprised" emotion, to simulate that Mini was not expecting that response. After executing the *Surprise* gesture, the robot gave the last explanation, and engaged in one last dialogue with the user. The interaction ended with Mini summarizing the results of the game, before bidding the user farewell.

4.6.3.4 Conditions

Four conditions have been considered for evaluating the effect of profile-based modulation:

1. **Neutral:** Under this condition, the robot cannot display neither emotions nor moods. The modulation strategy was removed from the players, so the robot always displays the neutral state, and the emotional expressions sent by the emotion display module were replaced by non-affective versions that only included the utterances, in a neutral voice.

2. **Punctual-emotions:** Under this condition, the robot cannot display emotions or moods, but the emotional expressions were added back to the emotion display module.

3. **Emotion-mood:** Under this condition, the robot is able to display moods and the emotional expressions. This was achieved by modifying the Players so they would ignore emotions, and pay attention only to changes in the robot's mood.

4. **Full affect:** Under this condition, the robot is able to display emotional expressions and to modulate its expressiveness to represent both emotions and moods, including the decay of an emotion's intensity. The decay of each emotion was adapted so there is enough time to show how the robot's expressiveness changes from the moment the emotion is at is highest, to going back to displaying only mood.

Figure 4.20 shows the different affect states that the robot shows during the entire interaction, as well as the conditions under which those states are conveyed.

4.6.3.5 Questionnaire

The questionnaire developed for this evaluation is similar to the one that was used in the first one. The first section includes the questions oriented to retrieve demographic information from the participants, the second section included the video that shows the interaction that will be evaluated, and the third section presents the participants with the items from the RoSAS questionnaire. Because

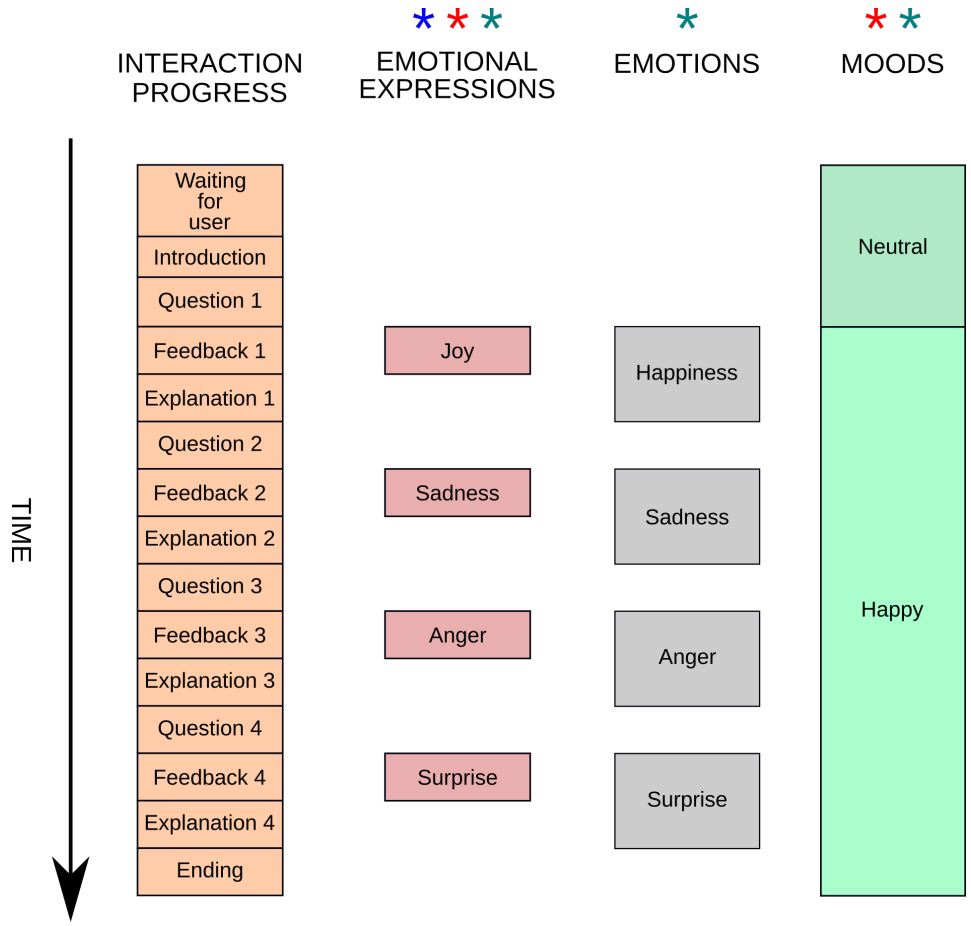


Figure 4.20: Affect states displayed during the interaction depicted in this experiment. Asterisks over each column indicate under which conditions is each type of affect state displayed. Green asterisk refers to the *punctual-emotions* condition. The red asterisk refers to the *emotion-mood* condition. Finally, the blue asterisk refers to the *full affect* condition.

these items are about generic social attributes that can be displayed by a robot, an extra section was added to explicitly evaluate the participants' perception of the robot's mood. In this section, participants were first asked if they were able to perceive any mood in the robot, and if their answer was affirmative, then were asked to specify which one. This was done through a text box that allowed each participant to provide the answer in their own words. Four versions of the questionnaire were prepared, each one with the video associated to one of the four conditions. Participants were redirected randomly to one of the questionnaires, in order to balance the number of participants in each condition.

4.6.3.6 Hypotheses considered

Because the goal of both subjective evaluations conducted is to prove if the modulation methods can be used to improve the perception that users have of the robot, the hypotheses considered for this second evaluation are the same four tested in the first one:

- **H1:** The modulation of the robot's expressiveness will affect the perception that the participants have of the robot.
- **H2:** The participants' perception of *warmth* will be directly related to the affect expression ($warmth_{neutral} < warmth_{punctual-emotions} < warmth_{emotion-mood} < warmth_{fullaffect}$).
- **H3:** The participants' perception of *discomfort* will be inversely related to the affect expression ($discomfort_{neutral} > discomfort_{punctual-emotions} > discomfort_{emotion-mood} > discomfort_{full\_affect}$).
- **H4:** The participants' perception of *competence* will show no significant differences between all four conditions.

4.6.3.7 Pre-evaluation stage

Although the goal of the experiment was to evaluate the effect of the modulation on the overall appearance of the robot, there is a previous step that had to be performed beforehand: ensuring that the modulation can be perceived correctly and has the desired effect. This means that the participants have to be able to recognize the emotions and moods that the robot conveys, and also that the emotional expressions designed are perceived as expected. In order to do this, an independent evaluation was conducted among participants that were not invited to take part in the study, so the results would not be skewed.

In this pre-evaluation, the robot was recorded performing the four emotional expressions designed, and then performing a generic gesture under all affect states used in the experiment (neutral and happy for the mood, and all four emotions). This resulted in ten separate videos. In an online survey, participants watched all ten videos, and then were asked to assign one mood or one emotion to the robot in each video (participants were told if they had to assign mood or emotion). This was a two stage process, where first they were allowed to write whichever affect state they considered the robot was displaying, and then were presented with all the affect states the robot can display and had to choose one.

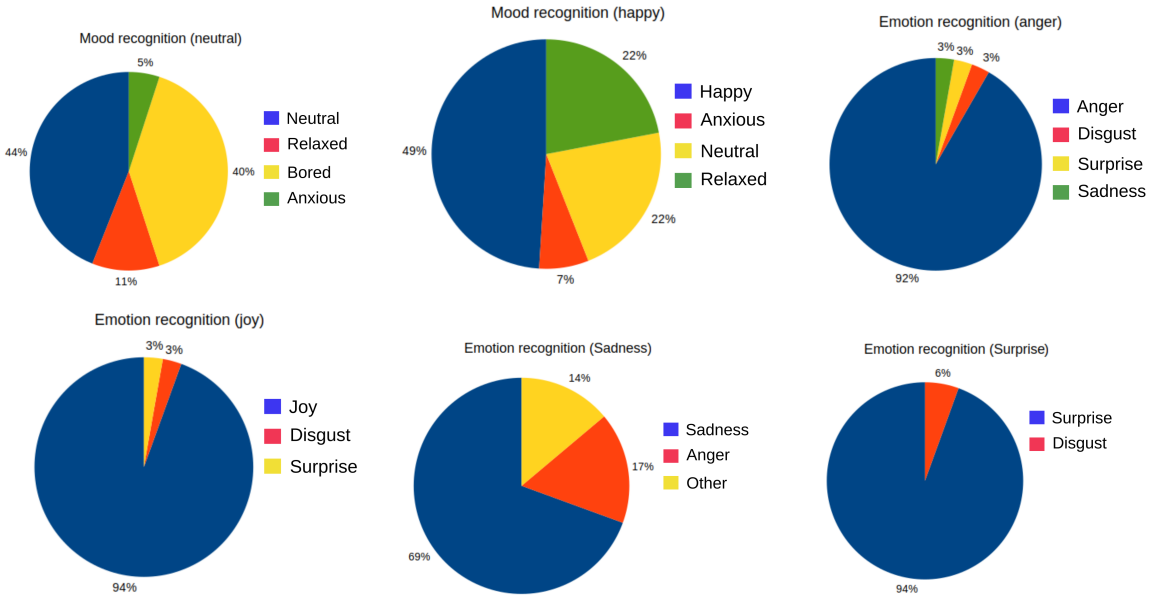


Figure 4.21: Recognition rates for the different moods and emotions displayed by Mini.

55 participants took part in the pre-evaluation. Among all affect states, the worst recognition results were obtained for the videos where the robot was displaying mood. This is understandable, as mood is less intense, and thus a change in mood has a smaller effect over the robot's expressiveness. When presented with a limited set of options, the correct mood was the most selected option (44% for *neutral* and 49% for happiness). In the open question, the neutral mood was described as *serious* by the majority of participants (44%), with *neutral* being the second option (27%), and the *happy* mood was correctly recognized by the majority of participants (36%). The results for emotion recognition are much more positive. Because the first design showed a low recognition rate, the emotion modulation profile was modified and a second questionnaire was developed and completed by 36 participants. *Happiness*, *anger*, and *surprise* show recognition rates of over 90% when the participants were presented with a closed set of options. For sadness, the recognition rate was significantly lower (69.44%), but still the most selected option. In the free text evaluation, the four emotions were still the most common options provided by the participants (58.3% for anger, 69.4% for happiness, 41.67% for sadness, and 50% for surprise). Figure 4.21 shows a compilation of these results. Finally, the emotional expressions were correctly recognized by more than the 70% of participants. Specifically, the recognition rate was 78.2% for *anger*, 87.3% for *happiness*, 70.9% for *sadness*, and 80% for *surprise*. Similar to the other affect states, the recognition rate was lower when the participants were allowed to provide an open answer. Still, the majority of participants still recognized correctly all the expressions. These results can be seen in Figure 4.22.

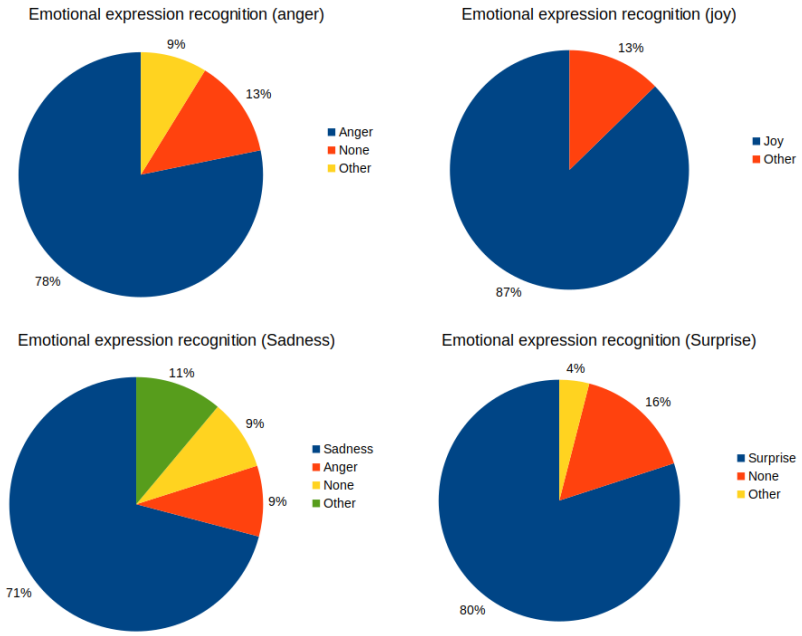


Figure 4.22: Recognition rates for the different emotional expressions designed for Mini.

Overall, the results obtained for this pre-evaluation can be considered as satisfactory. Emotions and emotional expressions were recognized by a significant majority of participants. The differences on recognition rate between open and closed questions can be explained with the fact that only terms that were close enough to the correct answer were considered as correct. For example, although *serious* could be considered an appropriate description for the *neutral* mood, it was considered as a different answer, which affected the results. The low recognition rates for the robot's mood were expected, due to the inherent difficulty that its recognition has without a proper context. Nevertheless, the majority of participants were still able to correctly recognize it.

4.6.3.8 Participants

Participants were contacted using the same methods described for the first subjective evaluation. In total, 166 participants took part in the experiment, 38 assigned to the *neutral* condition, 43 to the *punctual-emotions* condition, 40 to the *emotion-mood* condition, and 45 to the *full affect* condition. The split between male and female participants was 44%-56%. 12% of the participants had basic education, 16% were high school graduates, 11% had professional formation, 43% were college graduates, 15% had a masters degree, and 3% had a PhD. 86% of participants reported having medium or higher familiarity with technology, while only the 37% reported having a mid, mid-high or high familiarity with robotics. Regarding their interest on interacting and owning a robot, 82%

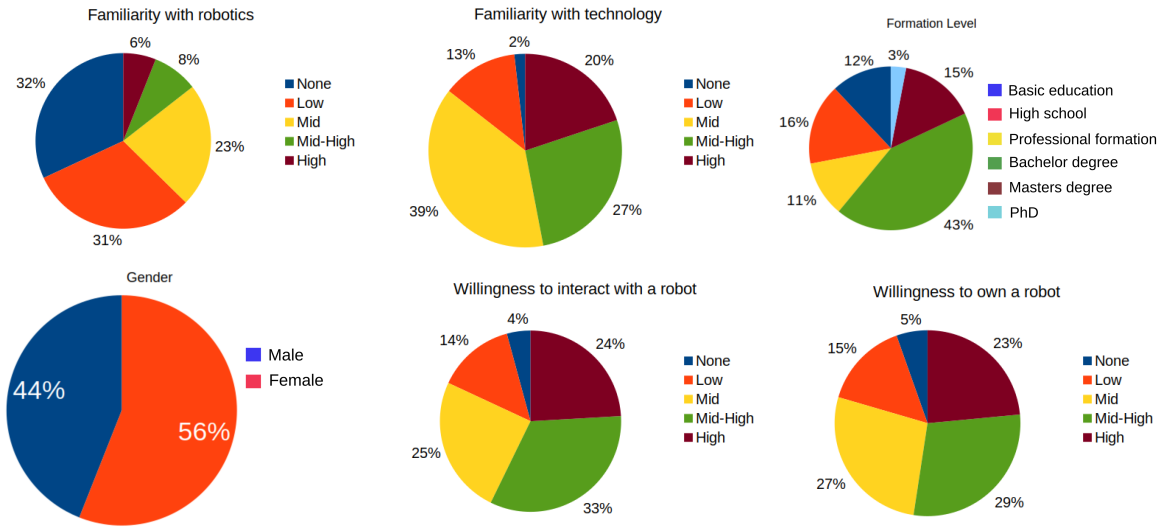


Figure 4.23: Demographic analysis of the participants in the experiment.

reported at least a medium interest in interacting, and 80% reported at least a medium interest in owning one. Figure 4.23 shows a summary of this demographic analysis.

4.6.3.9 Results

The 166 responses were analysed first in search for any duplicate answers. Next, the three dimensions of the RoSAS questionnaire were computed for each response, and the average value for each of these three items were computed for the whole set of data. Normality tests were conducted for the three dimensions. The results of the Saphiro-Wilk test show that the *warmth* ratings for all conditions did not deviate significantly from normal ($W_{neutral}(38) = 0.977, p = 0.599$, $W_{punctual-emotions}(43) = 0.968, p = 0.276$, $W_{emotion-mood}(40) = 0.974, p = 0.491$, $W_{full\_affect}(45) = 0.982, p = 0.707$), while the same cannot be said for *competence* and *discomfort*. Trying to achieve normality, a square root transformation was applied to both the *competence* and *discomfort* dimensions, resulting on the creation of the *sqrt_competence* and *sqrt_discomfort* variables, respectively. The Saphiro-Wilks test was conducted again for both variables, and the results show that the transformation was enough to achieve normality for the *sqrt_discomfort* variable ($W_{neutral}(38) = 0.967, p = 0.317$, $W_{punctual-emotions}(43) = 0.959, p = 0.127$, $W_{emotion-mood}(40) = 0.945, p = 0.052$, $W_{full\_affect}(45) = 0.956, p = 0.084$), but not for the *sqrt_competence*. Nevertheless, an analysis of the normal Q-Q plots for the *competence* variable does not reveal any major problems with kurtosis or skew, and the data in the graph tends to group around the diagonal line, which represents the perfect normal distribution. Thus, normality is going to be

also assumed for *competence*. Figure 4.24 shows the Q-Q plots for the *competence* dimension under all four conditions.

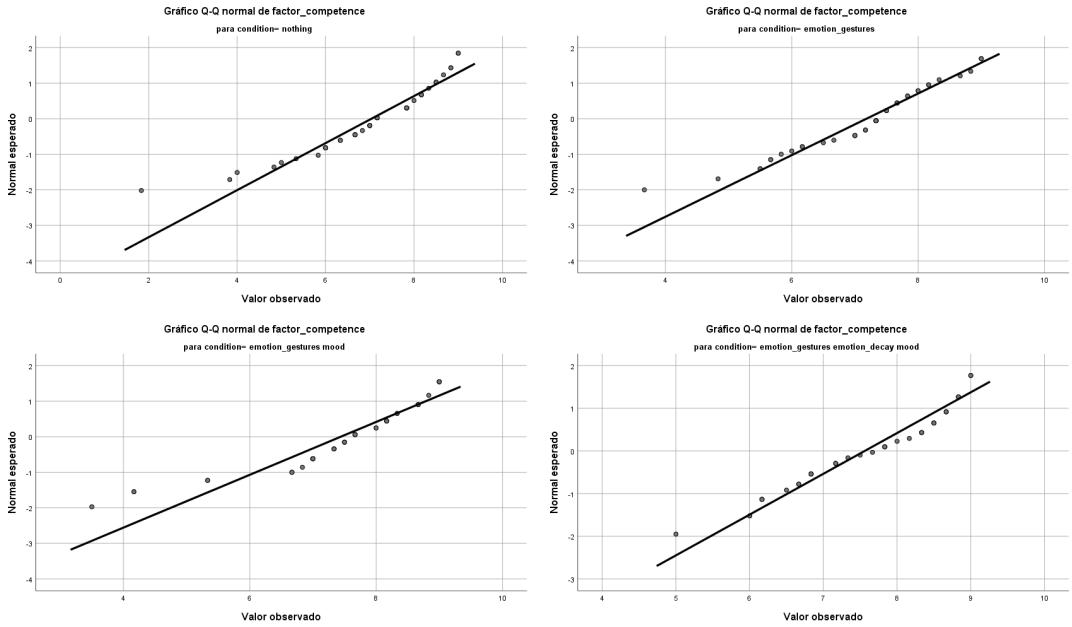


Figure 4.24: Q-Q graphics for the *competence* dimension under the *neutral* (top left), *punctual-emotions* (top right), *emotion-mood* (bottom left), and *full_affect* (bottom right) conditions. The line represents a perfect normal distribution.

After the normality assumption was checked, descriptive statistics were computed for all three variables (*warmth*, *competence*, and *discomfort*). The results for the *warmth* factor shows the highest rating for the *emotion-mood* condition, followed by the *neutral* condition, while the lowest rating was observed for the *full affect* condition. Regarding the *competence* factor, the highest rating was observed for the *neutral* condition, while the lowest was observed for the *full affect* condition. Finally, for the *discomfort* factor, the lowest rating was observed for the *neutral* condition, while the highest was observed for the *full affect* condition. In any case, none of the differences found between conditions for any of the three dependent variables proved to be significant. The results of the descriptive statistics can be seen in Figure 4.25.

The next step in the study was to use the four questions added to the demographic information section of the questionnaire as co-variables that will be controlled during the comparison of the dependent variables. The only co-variable that led to differences between conditions being found is the willingness to interact with a robot. First, an one-way ANCOVA was used to compare the ratings for *sqrt_discomfort* under all four conditions using the willingness to interact as co-variable.

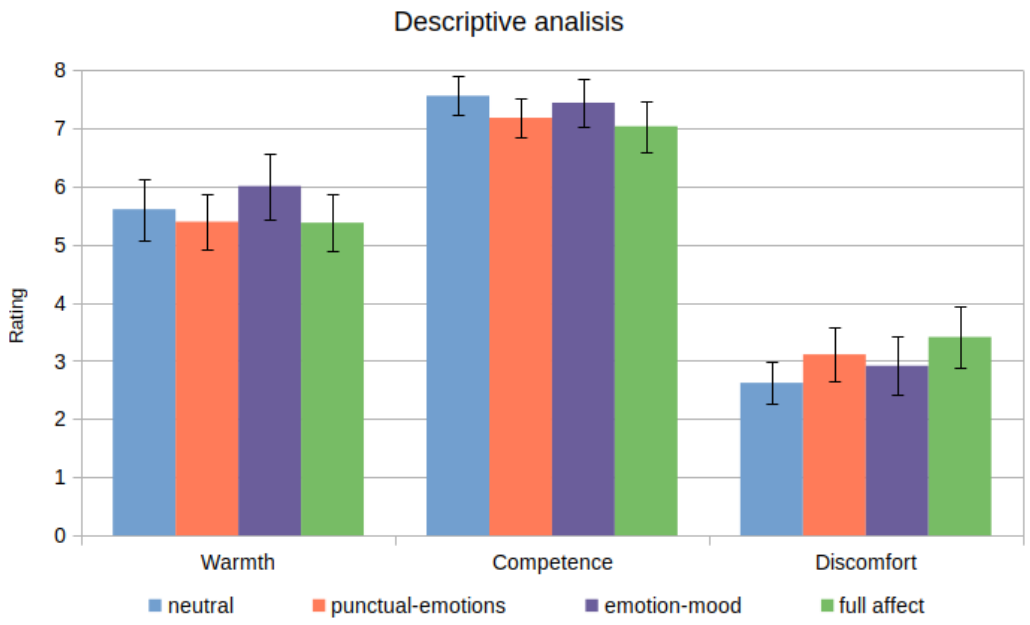


Figure 4.25: Ratings for the three sub-scales in the RoSAS questionnaire (warmth, competence, discomfort) for all four conditions. The bars represent the average value and the whiskers represent the 95% confidence intervals.

Normality assumptions were met, the result of the Levene's test carried out showed that homogeneity of variances can be assumed, and a Sidak correction was performed. The results show that there is a significant difference on the ratings for *sqrt_discomfort* ($F(4, 161) = 3.207, p = 0.014, \eta^2 = 0.74$). In order to understand which pairs of conditions are significantly different from each other, post-hoc tests were conducted. They revealed that the ratings for *sqrt_discomfort* under conditions *neutral* and *full affect* are significantly different ($p = 0.0443$). This means that the modulation of the robot's expressiveness using emotional expressions, intensity-based displays of emotion, and expression of mood lead to significantly higher ratings of discomfort when compared with a robot that is not able to express any affect state. Next, another one-way ANCOVA was conducted to study the variations on the ratings of *competence* while controlling the willingness that participants have to interact with a robot. In this case, the differences found were only marginally different ($F(4, 161) = 2.305, p = 0.061, \eta^2 = 0.54$). The results indicate that there is a direct relation between the willingness to interact with a robot and how competent it is perceived. Finally, no correlation was found between the ratings of *warmth* and how willing to interact with a robot were the participants.

4.6.3.10 Discussion

None of the differences found in the analysis of the results obtained were significant, and thus, no conclusions can be extracted from these results, beyond the fact that is necessary to conduct more trials to try to understand which could be the reason behind this lack of significant differences. With this results, hypotheses **H2** and **H3** are not validated, as there was no significant differences in the ratings of *warmth* and *discomfort*. **H1** can also be considered not to be validated, as there are only significant differences under a very particular condition (when the willingness to interact with a robot is used as a co-variable). Finally, the only hypothesis validated by the results is **H4** as there are no significant differences in the rating of *competence* among conditions.

There is a series of possible reasons that explain why this experiment did not show any significant results. First, Mini was designed to look cheerful and friendly, with a squashy body, and a voice that is intended to be warm. This could suggest that the configuration used for the *neutral* condition could not been perceived as neutral by the participants (here I refer to the appearance of the robot during the entire interaction, and not to the neutral mood that was tested for a single expression in the pre-evaluation stage). However, a more thorough evaluation of the robot's *neutral* appearance should be conducted. The fact that the lowest rating of discomfort was observed for the *neutral* condition and the highest was observed for the *full affect* condition could be tied to the definition of the experiment. In the quiz game depicted in the videos, the robot changed its affect state after every question, and the explanations that the robot provided went on for approximately 1 minute. This means that the robot changed from maximum *happiness* to maximum *sadness*, and then to maximum *anger* and maximum *surprise* in 1 minute. On top of this, the emotional expressions and displays of affect were developed with the idea that they should have a high intensity, so they could be recognized in videos without problem. There is a possibility that all of this led to changes in affect display that were too extreme, and thus were perceived as being more abnormal. One last possible reason is the interaction selected for the evaluation. A quiz game where the robot plays the part of the host might not be the situation where one would expect a heightened emotional response, and thus the neutral configuration could be perceived as appropriate. A possible solution could be identifying a situation where the desired emotional reactions are more appropriate. In any case, further tests are required in order to identify how to improve the proposed approach for displaying internal states using modulation profiles.

4.7 Conclusions

The design of an expressiveness management approach is one of the core tasks that has to be completed for endowing a social robot with the ability to interact with users in a human-like way. This task can be divided on two components. On one hand, the main goal of an expressiveness management module is to ensure that the gestures performed allow the robot to achieve its communicative goals successfully, in a way that meets the expectations that humans might have. This means that the actions should be performed in a human-like manner (avoiding repetitive or mechanical interactions), obeying the temporal constraints of communication. On the other hand, expressions not only have the goal of conveying messages, but should also transmit the internal state (emotional and motivational) of the robot. Any approach designed to generate multimodal expressiveness for a robot has to take care of both aspects simultaneously. In this chapter, an approach for developing multimodal expressiveness for a social robot has been presented.

The proposed approach has proven to manage both aspects of communication, being able to convey complex multimodal messages while abiding by the rules that control interactions between humans (particularly, the temporal constrains involved in all dialogues), while at the same time providing the tools required to transform these multimodal expressions so they are able to convey the internal state of the robot while still being able to complete their communicative goal.

4.7.1 Contributions and achievements

The solution proposed for describing expressions for a social robot represents multimodal gestures using state machine-like structures. In this approach, each of the unimodal actions (e.g., utter a sentence, move a joint...) that compose an expression is controlled by a single state. Under this approach, developers can design expressions from a global point of view, instead of focusing on developing the actions for each interface separately. This allows to combine two methods for synchronizing multimodal actions: (i) use timestamps to indicate the moment in which each action has to start, or (ii) tie directly the beginning of one action to the completion of a previous one. In order to simplify the process of creating new expressions, the state machines are based on FlexBE, a framework for developing high level behaviours for robotics. Among other features, this framework allows developers to create the expressions using a graphical interface, which allows to incorporate to the design process any person with the expertise required for creating human-like gestures, regardless of his/her background on programming.

The second contribution that has to be highlighted is the development of the Expression Manager, the software architecture that controls the process of loading, configuring, and executing expressions. This module is divided in three main components: (i) the Expression Scheduler receives requests for executing expressions coming from the rest of the robot's software architecture and plans their execution, using a priority-based approach for deciding the order in which these expressions should be performed; (ii) the Expression Executor receives activation commands coming from the Scheduler, loads the corresponding expression, configures it, and ensures its correct execution; and (iii) the Interface Players receive all the different actions that the expression needs to perform, and format them into commands that can be processed by the robot's output interfaces.

In order to diminish the negative effects of using predefined expressions, the third main contribution of this chapter is the development of a series of strategies designed to generate and modify expressions in runtime. First, a template was developed to build expressions from a list of actions. This approach starts by building a tree that represents the connections between the different actions in the list (the order in which they have to be executed, and if any of the actions needs to wait on the completion of a previous one). This tree is then used to create the appropriate state machine. This frees developers from the need to model everything the robot needs to do as a predefined expression.

Second, three modulation strategies have been devised for increasing the variability of the robot's expressiveness, allow it to display internal states, and adapt the predefined expressions to the current context of the interaction. The first strategy allows to replace individual actions inside an expression during the process of loading and configuring it in the Expression Executor. With this, gestures can be adapted in runtime to better suit the interaction requirements (for example, replacing the utterance in a greeting gesture so it references the time of the day, or the name of the user). The next modulation strategy can modify the global behaviour of an expression with two control parameters: speed and amplitude. Each parameter will have a specific effect over the robot's interfaces (for example, speed will control the velocity of the motions or the prosody rate, while amplitude can be used to change the pitch of the voice or the intensity of the robot's LED). Changes in the internal state of the robot, or specific needs of the robot's software architecture can lead to changes in the values of these two parameters, which in turn will result in the modification of an expression's appearance. The final modulation strategy allows developers to create specific modulation profiles detailing the effect that each internal state has over the parameters that control the robot's expressiveness. For example, while a change on the value of the *speed* control parameter will result in changes in both the speed of the robot's motions and its prosody rate (either both are

increased or decreased), using the modulation profiles allows roboticists to specify individual effects for each parameter.

The approach presented has been evaluated through a series of experiments, both subjective and objective, in order to ensure that is able to cover both dimensions of communication (conveying messages and displaying internal states) in an appropriate manner. This evaluation shows that the expressiveness architecture has a reaction time under most conditions below the more strict threshold defined for a natural interaction, and below the maximum threshold considered under all conditions. The results also suggest that, while not being its main purpose, the expressiveness system could be also used to manage reactive behaviours at a speed comparable to that of a person, if the behaviour is not excessively complex. Regarding the use of the modulation strategies, results suggest that they are able to serve their purpose, that the states conveyed through the modification of the gestures are recognizable by users, and that a proper modulation strategy that adapts the expressions to the state of the robot and the circumstances of the interaction can improve the perception that they have of the robot.

4.7.2 Achievement of the proposed goals

The goals regarding the implementation of an approach for managing the expressiveness features of a social robot were presented in Section 4.1.3 of this manuscript. The main objective that had to be achieved is the development of a model that allows to represent multimodal expressions for a social robot. Due to the complexity of this goal, it was decoupled in a series of subgoals that would serve as milestones of the progress made. This section evaluates the degree of accomplishment of each subobjective:

- The first subgoal defined in the introduction of this chapter was related to the process for creating new expressions. They should be easy to craft, and require as little programming knowledge as possible, in order to allow roboticists to involve in the design process people without a technical background, but with knowledge on other areas of interest, like animation. This has been solved through the use of FlexBE for creating new expressions. It provides a graphical user interface where developers can drag blocks representing the different states that can be used to build an expression and connect them to specify the transitions. This eliminates the need for touching any code, simplifying the design of new gestures. The original version of FlexBE has been adapted so it fits better the requisites of the task at hand.
- The second subgoal required the implementation of modulation strategies that could be used to modify the expressions in order to adapt them to the context. This goal was achieved

through the implementation of three modulation techniques: (i) allow the replacement of actions in a expression with new values in runtime; (ii) the use of two control parameters (speed and amplitude) to modify the overall behaviour of the expressions; and (iii) the definition of modulation profiles tied to specific states of the robot that allows an individual control over each parameter of an interface. The combination of the three proposed methods has been deemed enough for the situations that have been identified so far (for example, the expression of affect states through the modulation of the robot's expressiveness).

- The previous subgoal was extended with a second objective related to the users being able to properly recognize the different states that the robot might try to express using the modulation techniques implemented. This means that the modulation not only has to create a perceivable change in the robot's expressiveness, but the control of the modulation has to be precise enough to generate recognizable states. The results extracted from the subjective tests conducted for evaluating the Expression Manager suggest that this goal has been achieved successfully. In particular, a preliminary test was conducted during the experiment that studied the effect that different levels of affect display had over how users perceived the robot. This test showed that both the emotions and moods displayed using the profile-based modulation were recognized correctly by the majority of participants. Also, although no particular internal state was expressed through the adaptation of the speed and amplitude parameters, or the replacement of the communicative actions in an expression, the results of the second experiment suggest that these two methods can be used to improve the perception of *warmth* coming from the robot.

- The next subgoal defined that the model used for representing expressions had to allow any combination of communication modalities, while ensuring that this does not lead to a potential conflict between interfaces. The use of a state machine-like structure where all the communicative actions are sent to the Interface Players through one of the expression's states allows to combine these individual actions in any order required, or even perform them on parallel. While the developer is the one that has to ensure that the structure of the gesture does not causes conflicts in the use of the Interface Players, problems between expressions are handled by the Expression Manager itself. A priority system was implemented to solve any potential conflict during the planning stage. As a result of this process, expressions can be executed immediately, stored to be executed as soon as possible, or even discarded (if the priority is too low).

- As long as two expressions do not require the use of the same interfaces, they should be executed in parallel. This increments the expressiveness of the robot, as expressions that use limited

channels of communication could be enhanced with other less important expressions. An example of these less important expressions are behaviours that, while not having a particular communicative goal, can be used to endow the robot with a liveliness appearance. Thus, one of the subgoals defined for this research was the addition of the functionality required for executing multiple expressions at the same time. The completion of this objective required the modification of the FlexBE architecture. FlexBE was originally meant for managing high-level robot behaviours, and as such, was designed to control one behaviour at a time. Through the proper adjustments in FlexBE's source code, multi-behaviour execution was achieved.

- The expressiveness that a social robot needs to be able to display could involve more than just individual actions, or even multimodal expressions. Some situations could require the activation of more complex behaviours that need to be performed for extended periods of time. An example could be the use of a face tracking behaviour, where the robot follows the face of the user with whom it is interacting. Although the development of this type of complex behaviours falls outside of the scope for this dissertation, one of the goals defined was to expand the Expression Manager with the mechanisms required to integrate these behaviours in the robot's expressiveness. This goal was successfully completed through the implementation of a series of FlexBE states: one for sending the signal required to activate a complex behaviour, another for controlling that the behaviour is being performed correctly, and a third one to deactivate the behaviour. These three states can be connected sequentially to create an expression that follows the same structure that the rest of the robot's gestures, and that can be used to establish a communication between the robot's expressiveness and external modules that implement these complex behaviours.

- Finally, the last subgoal that had to be accomplished was related to the performance of the Expression Manager. As stated in several parts of this manuscript, there are temporal constraints in human communication that have to be met. For example, if messages are not conveyed in under a certain amount of time, they could loose their meaning. In Sections 3.6.2.2 and 4.6.1.2, it was stated that the robot should be able to perform communicative actions in under 1 second, with the maximum delay being fixed at 2 seconds, in order to be compliant with the *"Two Second" rule*. The objective evaluation of the Expression Manager showed that, even under the worst conditions observed for the delay introduced by the HRI Manager, all communicative actions that involved the use of a single expressions were performed in under 1 second. For a complex expression involving multiple actions through different interfaces, the delay observed was slightly above the 1 second threshold, but well under 2 seconds, which was deemed to be

acceptable. Thus, although there is room for optimizing the performance of the system, it can be considered that the objective was completed successfully.

4.7.3 Limitations of the system and future lines of work

While the evaluation of the objectives that were defined for this chapter of the thesis suggests that the implementation of the new expressiveness architecture was completed successfully, there are still some areas that could benefit from extra developments. Based on this, a series of future lines of work have been identified:

- Currently, the approach presented designs all of its expressions using FlexBE, a framework designed for controlling high-level behaviours in robotics applications, which is built on top of SMACH, a library for developing hierarchical state machines in Python. While this solution proved to be effective for representing the expressiveness of a social robot, it might not be the most efficient. First, FlexBE was developed with high-level behaviours in mind, where only a single state machine would be running at a time, during the entire life of the robot. It was also designed with teleoperation in mind, where the graphical interface used to design the behaviours can also be used to control them. Second, the temporal analysis of the proposed expressiveness architecture showed some weak points in the process of executing expressions. In particular, the main bottleneck was found at the beginning of the execution process (the time that passes from the moment the execution method of the expression is called, until the first action is sent to the corresponding Interface Player). While under most conditions this delay is not enough to cause the loss of the message's meaning, the results pointed out that this delay is proportional to the complexity of the expression being executed. Third, FlexBE's interface provides a series of functionalities that are not required for creating expressions for a robot. Also, because all expressions follow a common structure, many parameters that have to be defined to create a FlexBE state machine will have the same value. Thus, automating the process of completing the values for these common parameters reduces the complexity of, and the time required for, creating new expressions. While using FlexBE for building expressions proved to cover all the needs identified for an expressiveness system, it could be positive to replace this framework with another approach that better suits the needs of the application at hand (the design of multimodal expressions for social robots). This would also imply to develop/find a new graphical interface for designing expressions.

- The current method is able to enhance the expressiveness of a robot by allowing the execution of multiple expressions at once. For example, while one expression can be used for achieving the communicative goal imposed by the current interaction, other expressions could be performed

for displaying the state of the robot, or simply to increase its appearance of being alive. Currently, when evaluating if two expressions can be performed simultaneously, the proposed approach only focuses on the interfaces that each expression will use. If there are no coincidences, then both expressions are performed. This could lead to a potential problem if two expressions that try to achieve opposite communicative goals are requested at the same time. In the current integration of this expressiveness approach, this is not a possibility, as the rest of the architecture ensures that this cannot happen. Regardless, it would be beneficial to develop an improved method that ensures that two expressions will only be performed simultaneously if both have compatible communicative goals.

- The developed approach uses a predefined library of multimodal gestures to generate the robot's expressiveness. As it has been already mentioned in this section, this can lead to repetitive or mechanical interactions if the size of the expression library is too small, or make it hard to manage if it is too big. The Expression Manager addresses this approach by accepting a list of actions and transforming it into an expression in runtime. But this solution still requires that the module of the robot's architecture that needs to communicate a message specifies all the actions that have to be included in the expression. Thus, it would be interesting to endow the Expression Manager with a method for automatically creating new expressions in runtime, based on contextual information and the communicative goal that has to be achieved. For example, if a user approaches the robot, the communicative goal behind the next robot's action could be greeting this user. This goal, along with contextual information, like the time of day, or the name of the user (if its known), could be used to create the appropriate expression. This would reduce the complexity of creating new interactions between the robot and a user, as developers could focus exclusively on specifying the goal of the robot's messages and leave the generation of the expression itself to the discretion of the Expression Manager.

- One of the modulation strategies developed proposes the use of two control parameters to modify the appearance of a particular expression: speed and amplitude. Each parameter can have a limited amount of settings, and has a specific effect on each of the robot's interfaces, which cannot be decoupled (for example, the speed parameter will always modify the speed of the robot's motions and also its prosody rate, but never one or the other). In the current version of the expressiveness architecture, the values for these two control parameters are specified in the request for executing an expression. This means that whichever module of the robot requests the expression has to decide which is the correct value for the control parameters (the *normal* value is assigned by default), and also that these values have to be added to every request. This issue could be solved with the implementation of a method that can retrieve information

about the robot's state and use it to infer the correct value that should be used for the control parameters. The Expression Manager should still allow the applications to specify values for these parameters (in case it is necessary for whatever reason), and only use the inferred values if none are provided with the execution request.

- The profile-based modulation strategy allows developers to define specific configurations for the different parameters of the robot's interfaces, according to the state of the robot. These profiles would include the configuration related to each possible state that the robot might experience. The Interface Players are the ones that load the profiles, keep track of the robot's state, and adapt the expressiveness accordingly. This strategy has been tested for conveying affect states (mood and emotions), and because of this, the mechanisms implemented in the Interface Players have been designed around these particular states. For example, right know, the Players will prioritize the expression of emotions over mood, and compute the values for each of the interfaces' parameters as the fusion of both affect states. This modulation strategy needs to be expanded to consider other possible states, and the algorithm for computing the values for the interface's parameters should be generalized so it becomes state-independent, this is, so it can fuse any two states that the robot might be experiencing.

CHAPTER 5

Liveliness in Social Robotics

5.1 Introduction

There is a motivation to be part of a group that pushes humans to forge social relationships, even under adverse conditions [165]. According to evidence, this need to belong plays a role on shaping emotion and cognition. For example, the status of the social bonds that a person establishes is going to have an effect on that person's emotional state (forging bonds has a positive effect, while situations in which a bond is in danger have a negative effect). Also, a person devotes an important part of cognitive resources to the creation and maintenance of these social bonds, and how he/she process information coming from another person is going to change depending on the existence of a shared bond. Humans create relationships with each other through social interactions. But for a person to engage in such interactions, it is required that he/she can recognize the other participant as being capable of establishing a mental connection [166]. Thus, for a robot to become an effective social agent, it has to be recognized as an appropriate interaction partner by the humans in its environment.

Biophilia can be defined as the hypothetical tendency that humans have to interact and affiliate with other life forms. First proposed by philosopher and psychologist Erich Fromm in his book *The heart of a man: Its genius for Good and Evil* [19], published in 1964, the term was popularized by Edward O. Wilson with his book *Biophilia*, published in 1984. In this book, biophilia is defined as *"the urge to affiliate with other forms of life"*. The original work presented by Wilson was later complemented by *The Biophilia Hypothesis* [167], a book that compiles the work of 20 researchers that examined how the concept of biophilia can be applied to different contexts.

Along the years, there has been a large body of evidence both in support and against the biophilia hypothesis. In [168], Kahn presented a review of some of the works in support of the biophilia, addressed some of the concerns arisen by this hypothesis, and also presented a discussion about structural-developmental studies conducted by him and his colleagues. Another example is the work of Ulrich and Lunden [169], which showed that people exposed to nature images during a surgery experienced lower levels of anxiety after the procedure. Regarding the criticism that biophilia attracted, Joye and De Block [170] contended the idea of integrating the feelings for life in an evolutionary psychology framework. Another concern that is raised is the existence of biophobia (the fear of nature). Regardless of the extent to which the biophilia hypothesis can be defended from all the criticism that surrounds it, what seems clear, based on all the evidence, is that humans do tend to feel affiliations towards life forms. Applied to the design of a social robot, it could be argued that, even if these affiliations are not the driving force of human behaviour, giving the robot a liveliness appearance could help it to create a relationship with users. This idea of a liveliness appearance will be related to the concept of animacy.

Animacy can be defined as *the state of being alive and animate* [171]. While some authors argue that animacy has to be understood as a binary concept (an entity is either animate or inanimate), others argue that it is a continuous scale [172]. Animated beings display 4 key features [173]: (i) they have the ability to move on their own; (ii) they can grow and reproduce; (iii) they are able to maintain a knowledge base, perceive their surroundings, generate and display affect, learn new concepts, and think; and (iv) they are composed of biological structures that allow reproduction and also maintain life. Inanimate entities, on the other hand, do not share any of these features. The current robotic technology makes impossible for a robot to exhibit the features ii and iv, as they cannot grow or reproduce, but they are able to exhibit the other two.

Research shows that animacy is a key feature for a robot that has to be considered a social actor. For example, Csibra et al. [174] conducted several studies to examinate the idea that humans attribute intentions, beliefs, desires, and purposes to non-human entities (including artificial elements, like a robot) if these entities have the ability of moving on their own. Their results show that the participants were able to perceive the actions of the entities as being goal-directed, even without being able to perceive the source of said actions. Besides the fact that animacy is a requirement for being consider a social actor, there are other advantages for a robot that displays a liveliness appearance. For example, experiments conducted by Bugaiska et al. [175] pointed towards the idea that stimuli coming from animate sources have a higher processing priority. Bartneck et al. [23] used a variation of the Turing Triage Test to study robots' animacy. Their results suggest that it would seem desirable to use facial

expressions as a way to attract attention on the robot, and also that a good solution for achieving an animate appearance is a combination of face animations and human features that use smooth gestures.

The review conducted in this section suggests the importance of the robot being able to transmit the idea that it has a mind of its own, with intentions and desires. The task of endowing a robot with this feature has a cognitive dimension, involving the creation of these intentions, desires, and beliefs, and an expressive dimension, represented by the behaviours that the robot can perform. The latter will be the focus of this chapter. One of the solutions proposed in this manuscript is the use of co-speech gestures (non-verbal expressions that accompany a verbal message) for generating this liveliness appearance. Thus, besides considering the factors that make a robot look alive, it has to be considered also the relationship that exists between speech and gestures in human communication. In face-to-face interactions, participants often use co-speech gestures on top of the verbal modality, while listeners have to integrate gestures and other visual cues (for example, facial expressions or lip motions) with the speech in order to obtain the whole meaning of the message being conveyed [176]. Several factors suggest a tight connection between gesture and speech production [177], including the fact that both modalities communicate congruent information, and the performance of a gesture can be tied to a specific point in the speech, or the fact that we might still use co-speech gestures even if the listener could not see them, or that speech fluency suffers if the speaker cannot perform said gestures. However, other authors defend that the generation of co-speech gestures is not always tied to speech production [178].

In gesture-speech integration, although both speech and gestures share meaning and co-occur, they can also convey separate information [179]. Co-speech gestures can be broadly categorized into one of two classes. On one hand, there are gestures that have to follow a set of rules in order to have meaning. They are known as emblematic gestures, also called quotable gestures. Their form and metaphoric interpretation are defined by conventions. An example of this would be a hand gesture where the tips of the thumb and the index finger are touching, forming a circle, while the other fingers are extended. This gesture is generally understood as *"OK"*. It has to be mentioned that these gestures do not necessarily replace the speech, but instead add more meaning to it. The second class of gestures, known as non conventional, are those that are generated automatically during speech and do not follow any rules. This type of gestures can be considered part of language, although without being redundant, and do not have a standard form. They are connected to specific segments of speech, and have to be precisely timed to coincide with them. An example of non-conventional gesture would be mimicking breaking something with the hands while explaining how an object broke. In [179], McNeill recognizes four categories of non-conventional gestures: point, beat, iconic

and metaphoric. According to this work, the relationship between speech and gesture can be seen as the integration of two contrasting semiotic systems that share a common core idea, but express it differently.

There have been multiple theories for how speech and gestures are interconnected. Growth point theory of co-speech gesture generation [179] proposes that speech and gestures have their origin on the same representation, this is, an idea unit that combines verbal and non-verbal component. The sketch model of co-speech gesture generation [180] proposes that both the speech and gestures are generated from the same communicative intention. During the conceptualization phase, this communicative intention is realized by generating the appropriate speech representation of said intention, and the imagistic representation of gesture contents. The lexical retrieval hypothesis [181] proposes that co-speech gestures are generated during speech production, specifically during the formulating phase, where the semantic features of certain lexical items are used to generate the gestures. Regarding the structure of each individual gesture, a standard convention separates the motion in preparatory, stroke, and retraction stages, although some authors presented variations of this structure [182]. The speech-gesture synchronization approach that is proposed in this chapter of the thesis takes ideas of the sketch model and the lexical retrieval hypothesis for connecting speech and gestures.

The conclusions that can be extracted from the research presented above is that there are multiple factors that play a role on conveying animacy, including the external appearance, the actions that the robot performs, and the use of non-verbal communication, among others. In this dissertation, the problem of how to increase the animacy of a social robot is tackled through two different methods. The first solution proposed generates random actions for each of the robot's communicative interfaces. Because the actions generated by this method can be run alongside the robot's speech, but the information contained in the verbal message is not used during the generation process, it is important that these actions are generic, in order for them to not contradict the meaning of the utterances. This can be seen as a drawback, which the second proposed approach tries to overcome.

The second method seeks to generate co-speech non verbal behaviours. There are two main approaches for solving this problem. The first solution involves combining manually verbal and non-verbal components into multimodal expressions. This has the advantage of allowing developers to carefully design each of the different actions that will conform the expression, although it is time consuming, and can make interactions feel repetitive, if the amount of expressions designed is too low. The second approach relies on models that generate the expressions automatically. This

solution simplifies the process of adding new dialogues to the robot's repertoire, although the expressions generated automatically can end up being more generic. Authors have tried to solve this disadvantage by adding different constraints to the generation process that allows the system to adapt the expressions generated to different external factors (for example, switch the style of the expressions created based on the identity of the human speaker). In this chapter, the solution proposed combines elements of both approaches. The problem of co-speech gesture generation has been framed as a gesture prediction and synchronization task. The non-verbal behaviours have been manually designed to represent different communicative intentions, while a model has been trained to automatically select the most appropriate behaviours given the verbal information that has to be conveyed. This allows developers to add new verbal messages without the need for designing also the non-verbal information, and the non-verbal expressions can be carefully designed around the communicative intentions that the robot might display.

The proposed solution defines a set of gesture labels that represent all the possible gesture types that the robot can perform. In this case, each label represents the semantic value of the gesture that should be performed alongside the speech. Examples of these labels include *greet*, *question*, or *express enthusiasm*. A model is trained for selecting the sequence of labels that better match the transcription of the speech. This is done by considering both the actual words, and their part of speech function, as well as the communicative intention of each speech chunk (for example, greet the user, thank him/her for something, ask a general question, or ask a question about the other speaker). For each possible gesture label, a series of non-verbal behaviours have been handcrafted to represent that particular meaning. The proposed model will predict the sequence of gesture labels that have to accompany the speech, and then will use these labels to select the correct behaviours and synchronize them to the speech. The resulting multimodal expression is then sent to the Expression Manager to be executed. This allows roboticist to use in their applications handcrafted multimodal gestures, specify a list of actions that have to be transformed into a multimodal expression in runtime, or define only verbal messages and let the proposed approach select the most appropriate non-verbal component. Figure 5.1 shows the software architecture described in Chapter 2. This chapter will focus on the modules highlighted in red.

5.1.1 Objectives

This chapter is devoted to study how to increase the animacy of a social robot. Previous approaches designed in the research group opted for tying each internal state of the robot to a set of expressions and just draw expressions randomly at a given rate. However, this approach ignores the deliberated

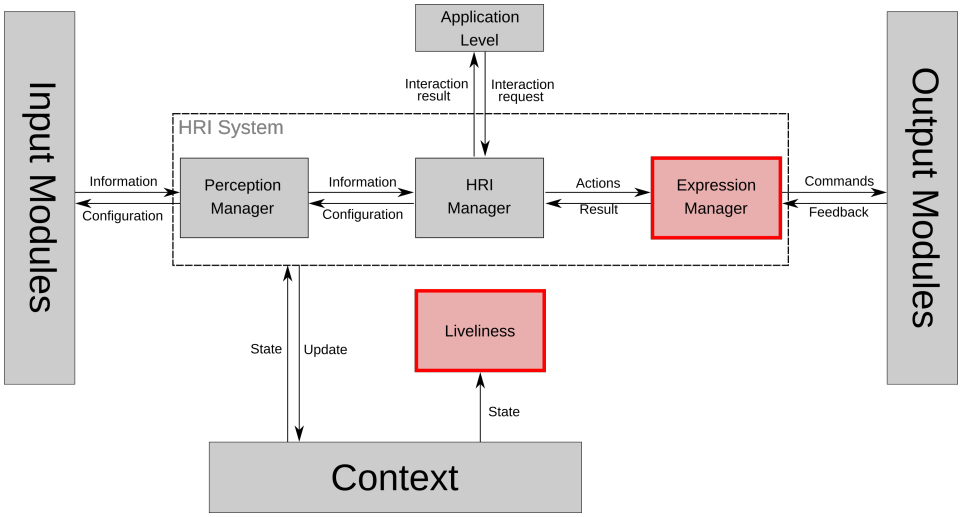


Figure 5.1: Diagram representing the software architecture described in Chapter 2. The work developed in this chapter of the dissertation focuses on the modules of the architecture highlighted in red.

actions that the robot might be performing at any given time. So, if one of the robot's applications try to convey a verbal message, and the random expression selected for enhancing the robot's liveliness appearance does not suit the information contained in the speech, it could be perceived as unnatural by the users. This thesis aims at **enhancing the liveliness appearance of the robot through the integration of two methods for displaying non-verbal, non-functional expressions**. This objective has been characterized in two main goals:

1. **The implementation of a method for generating unimodal actions (actions involving a single mode of communication) for each of the communicative modalities that the robot can use.**

2. **The development of a co-speech gesture prediction module for enriching the speech of the robot with appropriate, non-verbal expressions.**

Due to the complexity of these goals, a series of subgoals that can be used as milestones for achieving the final objectives have been defined. The first milestone is related to the method for generating unimodal actions, while the next five are related to the co-speech prediction method. Finally, the last subgoal is related to both methods:

1. The unimodal action generation method has to be able to adapt the actions created to the internal state of the robot.

2. The proposed solution for enhancing the robot's speech with non-verbal behaviours will rely on the speech's communicative intention for selecting the appropriate behaviours. Thus, two models have to be developed, one for predicting intentions based on the speech, and another for predicting the non-verbal behaviours that have to be used based on the speech and the output of the first model.

3. The appropriate datasets for training the co-speech prediction gesture system have to be generated. This involves creating two datasets, one for the model that predicts the communicative intentions of the utterance, and another for the model that predicts the non-verbal behaviours.

4. A method for synchronizing the selected non-verbal behaviours with the speech has to be developed. Because the length of the utterances and the number of gestures required is unknown and can change from utterance to utterance, the method needs to be able to handle the synchronization of a list of gestures with variable length.

5. The proposed co-speech gesture prediction system will have to select the appropriate behaviours from a library of predefined expressions. Gesture-speech synchronization has to be easily modifiable, so it can be adapted to the available gestures.

6. The co-speech gesture prediction module has to be easy to integrate in robotic platforms with different morphologies. In order to do this, gesture prediction has to be detached from gesture execution.

7. The proposed methods for enhancing the robot's animacy have to be integrated in the robot's software architecture, be able to work autonomously, and show a level of efficiency that allows for fluid, natural interactions.

5.1.2 Overview of the chapter

Here, the different sections that compose this chapter are described:

- Section 5.2: In this section, a review of co-speech gesture generation approaches is conducted. This analysis includes both works that either generate the non-verbal behaviour from scratch and those that select gestures from a library. Similar to the analysis conducted in previous chapters, all the works reviewed are compared based on a series of features that this dissertation defines as relevant. The analysis is completed with a comparison between the proposed model and the works reviewed, pointing out the similarities and differences.

- Section 5.3: Here, both the method for generating unimodal actions and the model for co-speech gesture prediction are presented. For the first method, its architecture is presented, along with its integration in the HRI System presented in Chapter 2. For the second method, this section presents the gesture prediction pipeline, the prediction correction mechanism, and the rule-based gesture synchronization approach. The co-speech gesture prediction model has been evaluated using mainstream machine learning metrics. This section also shows how this model has been integrated in a robotic platform.

- Section 5.4: This section introduces the experiments conducted in order to evaluate the proposed methods. First, an objective evaluation was conducted, which measured the time required by each method to generate their behaviours. Second, a case of use was presented to illustrate how both liveliness generation approaches are combined when the robot is performing one of its tasks.

- Section 5.5: This last section contains the conclusions extracted during the design and development of the proposed liveliness generation methods. This section includes a review of the goals that were proposed at the beginning of the development phase, and also highlights the most relevant contributions of the work presented in this chapter.

5.2 State of the Art

This section starts with an analysis of the effect that different non verbal communication interfaces have over the perception of animacy. Next, the analysis focuses on the generation of co-speech gestures, both for virtual agents and for robotic platforms. This review aims at endowing the reader with a general idea of the different solutions that can be implemented for enhancing the communication capabilities of an agent (either physical or virtual) with non-verbal behaviours that complement the verbal modality, in order to highlight the unique aspects of the work presented in this chapter.

5.2.1 Effect of non-verbal communication in the perception of animacy

There is an extensive body of work that studies how the modulation of different non-verbal interfaces influences how users evaluate the animacy of an individual. Most of the work revolves around two

types of actions: body motions and facial expressions. These are two of the most used communication modes in non-verbal expressions for robots. On top of this, robots can also be equipped with other communication modalities, like screens for displaying multimedia content (for example, images, videos, GIFs...), or coloured LED. This study will focus on the effect that non verbal behaviours have over the perception of animacy.

Regarding the study of how kinesics (body postures and motions) affects the perception of animacy, most of the work evaluated focuses on motion, and how the introduction of changes that seem deliberate in said motion can make even a basic shaped entity look animate. Santos et al. [183] studied how a proper parametrization of motions can influence the perception of animacy. In this work, animacy experience is described as the perception in an entity of motion patterns that would be attributed to an intentional agent. This animacy experience can lead to ascribe mind to a given agent. In their experiment, two different animations were used, one where one sphere moved in an horizontal trajectory, and another where two spheres were present, one in motion and the other static. Three variations were developed for the motion of the first sphere (the same were used in both videos). The dimensions of movement that were parametrized in this study are: (i) directionality, which represents the direction of the motion, as well as changes in said direction; (ii) Discontinuity, which can be understood as the existence of pauses in the motion; and (iii) responsiveness, which represents the reaction that the motion generates in the environment. Participants judged the motions in an animacy scale, ranging from *physical* to *personal*. In a follow-up experiment, the authors used only scenes that included two spheres to continue exploring the parameters that can modulate animacy. This was done by adjusting the modulation of the parameters used in the first experiment, as well as the motion variations. The results obtained from the studies show that the increase of animacy experience is influenced by the time the entity spends near a second entity, and also by the degree of interaction between said entities, although it seems that impressions of animacy are modelled by the combination of multiple variables. Chang and Troje [184] presented three experiments to assess animacy based on biological motion cues. The objective is to test the hypothesis that establishes that the perception of the animacy displayed by a entity is going to be influenced by the presence or absence of the visual invariants that signal direction and that are used to signal the presence of an animal. The stimuli used in these experiments were derivations of point-light sequences of a human, a cat, and a pigeon. The results of the experiments showed that, even though point-light displays do not have a coherent form, they still can display animacy. Because of the lack of coherence, this correlation must be related to how local motions are processed. A second conclusion extracted from the experiments is that this perception is shown to be orientation-specific. The second experiment showed that not being able to discriminate the direction of the motion leads to a decrease in the

degree of animacy perceived. Tremoulet and Feldman [185] demonstrated that an individual object can be perceived as being alive based on its motion pattern. The authors hypothesized that changes in speed and motion direction at the same time can influence a change in the animacy perception of the moving object. The environments were kept completely featureless to remove any possible effect that static objects in the environment could have. Besides the changes on speed and direction, three alignment conditions were considered: circular patch, rectangle aligned with the motion, and rectangle aligned always with the original direction of the motion. Participants observed the motions and categorized the objects as either alive or not. The results confirmed that animacy can be displayed in short motion paths if the direction and speed are altered, even without any explanation for these changes. The reason behind this can be that these variations in motion give the impression that the entity is able to control its own motion.

In the study of the relationship between facial expressions and animacy, authors have tried to find if any facial feature has a significantly bigger effect on animacy expression than the rest. They have also tried to understand the biological mechanisms involved in recognizing a face as belonging to a lively entity. Looser and Wheatley [186] studied the relation between animacy and facial expressions. Faces generate an indiscriminate effect of attracting attention. Among the faces detected, humans have to discriminate those that have a *mind* attached. In their work, the authors aimed at finding the point in which a face starts to appear alive, how gradual is this distinction, and also finding what perceptual information plays a bigger role in animacy attribution. Three experiments were performed, using images from faces that ranged from a mannequin to a real person. The results of these experiments suggest that faces need to be close to a human appearance, regarding its proportions, to be recognized as alive and having a mind. A second conclusion is that the attribution of animacy, opposite to realism, is not continuous, and instead seem to display categorical-perception effects. Also, the third experiment shows that the eyes have a disproportional importance in perceiving animacy. Koldewyn et al. [187] studied if animacy is a basic dimension of face perception, and if it is supported by a common neural mechanism across different categories of faces. Four experiments were conducted in this work. The first one evaluated if animacy shows adaptation that transfers between individuals, the next two evaluated if after-effects of animacy transfer between different face categories and across species, and the last one evaluated if animacy after-effects can be seen in inversion. In this context, adaptation after-effects can prove that the subset of neurons that process facial features are sensitive to a particular dimension (age, gender, etc...). The results of the experiments show reliable adaptation after-effects in how animacy is perceived in human faces. These after-effects were significantly weaker when the adapting stimulus was inverted. These results demonstrate a certain flexibility of animacy perception, and that this perception can be modulated over time by experience. The results also

show that animacy after-effects transfer across face categories (in this case, age groups), while they do not transfer across species. Regarding the cues that could drive these adaptation after-effects, the authors mention two types: surface-based and shape-based. Overall, the results suggest that animacy is a perceptual dimension of a face. Balas and Tonsager [188] presented evidence that suggests that the importance that eyes have over animacy perception is not a direct effect of eye appearance, but instead of the combination of this appearance with other facial features. In their work, the authors evaluated which are the facial cues that support the discrimination of animate and inanimate faces. 3D artificial faces were generated based on images of real persons. Contrast was used to manipulate both local and global features of the face in order to study what the real role of the eye area is on animacy perception, and also contrast chimeras (the independent manipulation of the eye region's polarity) in order to evaluate the importance of the eye region in face recognition. The results of the experiments suggest that both the eye region and the appearance of the rest of the face have an effect on animacy perception. It is important to mention that the eyebrows were not considered as part of the eye region when creating the contrast chimeras, and thus, its effect on animacy perception is not included in the eye area effect.

Finally, although the vast majority of authors have focused on the effect that motions and facial expressions have over animacy perception, there also researchers that have focused on robot-specific interfaces. For example, Rosenthal-von der Pütten et al. [189] presented a study where they compared the effects of human-like and robot-specific non-verbal behaviours on how the users perceived the robot. The robot-specific behaviour considered in the study is the use of coloured LED in the robot's eyes to display emotions, while the human-like expressions involved using body motions that are coherent with the situation in the study. Four conditions were defined: (i) the robot does not display any non-verbal behaviour; (ii) the robot displays only robot-specific behaviours; (iii) the robot displays only human-like behaviours; and (iv) the robot displays both human-like and robot-specific behaviours. Their results show that, regarding animacy, the addition of either human-like or robot-specific behaviours showed a significant effect on the users' perception. Human-like behaviours showed a bigger effect than robot-specific behaviours. Finally, the addition of robot-specific behaviours on top of human-like expressions did not show any significant differences.

Overall, the study conducted in this section showed that, in general, the use of non-verbal behaviours can help to enhance the animacy of a robot. One of the strategies that can be used for this is having the robot perform motions that seem intentional, and not caused by an external force. A second conclusion that can be extracted from this review is the importance that facial expressions have on animacy perception. In particular, the eyes might play an important role in shaping this perception,

according to some authors. Third, the work of Rosenthal-von der Pütten et al. [189] showed that even non human-like behaviours, like the use of coloured LED can help to convey a liveliness appearance. In this dissertation, one of the methods used for enhancing the robot's animacy focuses on generating unimodal actions through individual communicative channels, seeking to convey the idea that the robot is performing these behaviours intentionally, while the second method reinforces this appearance of intentionality of the robot's non-verbal actions by selecting those expressions that better suit the interaction, based on the communicative goals that the robot tries to achieve at any given time.

5.2.2 Comparison of co-speech gesture prediction/generation methods

A study of the applications for automatic generation of expressiveness for communicative agents shows that one of the applications that has attracted attention over the years is the extension of verbal communication with appropriate non-verbal behaviours. Besides serving a communicative purpose, the addition of these behaviours can help to endow the agent with a lively appearance that can improve the quality of the interactions with users. Although most of the works reviewed generate the motions automatically, a few works that couple the speech with handcrafted expressions have been included in the analysis, in order to provide a more complete picture of possible solutions to the co-speech gesture generation problem.

The first works reviewed focus on features that are present in the speech to select/generate the most appropriate behaviours. This involves both features extracted from the prosody and/or the transcript of the speech. In 2014, Chiu et al. [190] presented a machine learning-based approach to speech-gesture mapping. This approach divides the problem in two stages: (i) mapping the input audio features of the speech to a sequence of gesture annotations, and (ii) mapping the annotations to the appropriate motions. Conditional Random Fields (CRF) are used to generate the sequence of annotations, while the final motions are generated by Gaussian Process Latent Variable Models (GPLVMs).

The system starts by finding an appropriate mapping between audio features and a series of gesture annotations (there are only two possible labels: gesture and non-gesture, to simplify the learning process). Each gesture label is not only related to the input data, but also to the previous and following labels in the sequence. Once the annotations were generated, the system generates a sequence of continuous motions based on the labels, and links the motions one to another. This is done by learning a low-dimensional space that captures the dynamic aspects of the motions. The system encodes a set of motion samples into their low-dimensional representations. In order to

achieve this, the GPLVMs derive a manifold in which each point corresponds to a gesture frame. Gesture frames are selected in this manifold, and then an interpolation algorithm uses forward inference and backward inference functions to find the path between motions. Then, the trajectory found is mapped back to the original dimension through GPLVMs to obtain the final animation between gestures. One year later, Chiu et al. [191] proposed a new model for predicting co-verbal gestures: the Deep Conditional Neural Field (DCNF). This model provides a joint learning of deep neural networks and second order linear chain temporal contingency, where the former presents the advantage of mapping complex relations while the latter models the temporal coordination of the predicted sequence. The model receives a combination of content and prosody information about the speech, and predicts the most appropriate gestures.

The proposed approach takes advantage of gestural signs that represent the function and the form of co-verbal gestures. The dictionary of gestural signs was designed based on a theoretical analysis of the available literature on gestures, and then reduced by filtering those motions that showed low occurrences in motion data captured during face-to-face interactions. The DCNF model receives the text of the speech, part-of-speech tags for each word in the text, and prosodic features extracted from the speech. The networks in the model take this input for a given frame and forward it through the network to obtain one of the model parameters, while the undirected linear chain performs forward-backward belief propagation to compute the remaining parameters. The potential for each gesture sign is the weighted sum of the three obtained parameters, and the probability for said labels is defined as its normalized potential. In 2018, Ishi et al. [192] presented a method for generating hand gestures for android robots based on the speech. Based on an analysis of the relationship between occurrence of gestures and dialogue acts, the proposed method uses text, prosody, and dialogue act classification to generate the most appropriate gestures.

The proposed gesture generation method is divided in two phases: training and generation. In the training phase, words in the sentence are associated to concepts, which in turn are connected to gesture functions, and then to motions. The relationship between concepts, functions, and motions is modelled using conditional probabilities. Concepts are extracted from the words with WordNet. Probabilities between concepts and gesture functions are speaker independent, and computed using the full annotated dataset, while the association between functions and motions is tied to a specific speaker. Regarding the generation phase, two methods were proposed: text-based and prosody-based. The latter is in charge of the beat gestures, as they are usually tied to prosody peaks. In the text-based approach, the system extracts the sequence of concepts from the speech transcription, and then samples the gesture function based on the conditional probabilities. Based on the selected

function, a random gesture is extracted from the corresponding cluster. The stroke phase of the gesture is synchronized with the keyword that triggered it. The prosody-based approach searches for prosody peaks in samples taken from the utterance, and generates beat gestures for those peaks. Beat gestures can be superimposed to text-based gestures. Finally, the generated gestures can be discarded depending on the dialogue acts observed in the utterance. That same year, Hasegawa et al. [193] proposed a framework for generating body gestures to accompany speech based on a bi-directional Long-Short Term Memory (LSTM) neural network. The model is designed to learn the relationships between speech and gestures with backward and forward consistencies over a long period of time.

First, the system splits the speech into smaller audio chunks with fixed length, which are then converted to Mel Frequency Cepstrum Coefficients (MFCC) feature vectors. Next, each vector is passed to the LSTM network, alongside the vectors for the n previous and next audio chunks (the system uses silent audio vectors if there are less than n chunks after the current one). The network generates a frame of 3D positions for all the joints. Discontinuities between frames are corrected through a smoothing process that uses two different filters: a 1€ filter and a Moving Average filter. The former filters the low frequencies to eliminate small shimmering in low speed movements, while the latter smooths larger discontinuities using an average window over a certain range of steps. In order to train the system, a dataset was created through a semi-structured interview that paired speech audio and motion captured data. In 2019, Kucherenko et al. [194] presented a deep learning-based framework for speech-driven generation that incorporates representation learning. The proposed approach is divided in three steps: (i) learning a lower dimensional representation of the motions using an encoder-decoder architecture, (ii) training a new encoder to learn a mapping between the speech features and the lower dimensional representation, and (iii) connecting the original decoder with the new encoder.The baseline neural network architecture for this approach is the model proposed by Hasegawa et al. [193].

The original encoder-decoder architecture is modelled as a Denoising Autoencoder with one hidden layer. This network architecture is divided in two elements, an encoder that maps the input to the motion representation, and a decoder that reconstructs the original movement from that representation, both represented with Gated Recurrent Unit (GRU) neural networks. The system is forced to compute lower dimensional representations thanks to a bottleneck layer in the middle. Regarding the mapping from speech characteristics to motion representation, the proposed model replicates the architecture from the work of Hasegawa et al. [193], although in this case the output of the network is a low-dimensional representation of the motion. Three different features were tested as inputs to the encoder: MFCC features, spectrograms, and prosodic features. Kucherenko et al.

[112] introduced a year later a new model for generating random beat and semantic gestures together. The proposed model receives as inputs both the acoustic and semantic representations of the speech and generates expressions represented as a sequence of joint angle rotations.

In the proposed approach, audio is represented using log-power mel-spectogram features in a 64-dimensional vector, while the transcription of the speech is encoded using BERT. Each frame of both modalities is jointly encoded by a feed-forward network into a low-dimensional representation. Then, a sliding window of past and future speech features is passed over said representation in order to give more context. The encodings inside the window are concatenated and passed through several fully-connected layers. In order to ensure the continuity of the generated motions, predictions are fed back to the model. Also, the proposed system conditions the generation process on information from previous poses using FiLM conditioning, which generalizes regular concatenation. The model was trained without the autoregressive feature for the first epochs, so it does not converge to a static pose. Also in 2019, Yoon et al. [195] presented a learning-based co-speech gesture generation for humanoid robots that is trained from a dataset of TED talks. Similar to other models presented in this section, this approach relies on a encoder-decoder architecture for generating the motions. The system receives natural language text as inputs, and outputs frame-by-frame poses.

The speech text is encoded word by word into a sequence of one-hot vectors indicating the index of a given word in a dictionary. Regarding the poses, only the upper body is considered (arms, shoulders, neck, and head). These poses are converted into 10-D vectors by using Principal Component Analysis. The encoder is modelled as a GRU network, and is able to capture the context of the sentence while generating a lower-dimensional representation. The decoder is also modelled as a GRU network with pre and post-linear layers and a soft attention mechanism, for focusing in keywords in the sentence when generating the motions. The decoder takes into account the n previous poses generated to ensure the smoothness of the movement. The output of the encoder-decoder architecture is passed to a network that generates 3D stick figures from the 2D original predictions. Then, the joint configurations of the stick figure are directly copied into the robot. Speech and gestures are synchronized by dividing the speech in smaller audio chunks, and then generating motions for each chunk, with the same duration. That same year, Pérez-Mayos et al. [196] proposed a model that uses three different approaches for speech-gesture synchronization. The first approach starts by identifying keywords in the text connected to gestures in the database. Then, a motion sequence is computed by assigning emblematic gestures to the keywords and beat gestures to content words that have no emblematic gestures attached. Finally, the TTS module generates an event for each word that is about to be uttered, and executes the gestures attached.

The second approach aligns beat gestures to peaks in pitch. Each gesture has a pitch graph that is used to select the most appropriate expression given a utterance. The system requests the execution of the gestures using the time points associated to these peaks. The last approach combines both techniques, selecting emblematic gestures based on keywords in the speech and beat gestures based on pitch peaks. The former type will always have priority. Thus, the system will execute beat gestures until a emblematic gesture is requested, at which point the following beat gestures are discarded until the execution of the emblematic gesture is completed.

The authors also proposed a method for designing and storing human gestures to be reproduced by a humanoid robot. The gesture database in this work was designed by analysing common gestures that speakers in TED talks perform. Using a rule-based expert system, beat gestures are connected to the prosodic features of the speech, while emblematic gestures are related to the meaning of specific words. All gestures were manually designed. Also in 2019, Sadoughi et al. [197] introduced a model for generating meaningful behaviours based on the speech of a conversational agent. The model uses a dynamic Bayesian network with the addition of a discrete variable for constraining the behaviours based on the discourse function (for example, if it is a question) and prototypical behaviours (for example, head nods).

The baseline dynamic Bayesian Network is composed of observation and hidden variables. The former can be divided into *speech* nodes (represent the prosodic features of the speech) and *motion* nodes (represent either hand or head motions). The latter encode the state configuration between speech features and motions. The evidences fed to the *speech* node are used to predict the features of the gesture. The *motion* variable is computed with the Viterbi algorithm. The problem of states converging to the average position of behaviours is avoided with the Linde-Buzo-Gray vector quantization technique. After the network generates the motion trajectories, an extra processing step is applied in order to smooth the transitions between the key poses in the gesture. The baseline model is extended with the addition of new observational states representing the constraints applied to the gesture generation process. For each constraint, the model includes transition matrices that capture characteristic patterns associated to that constraint.

While the works presented above focus on features of the speech (either extracted from the prosody, the transcription of the speech, or both), other authors extended this information with other contextual information. One of the most extended approaches is using information about the other peer in the interaction (either his/her identity or his/her gestural style). For example, Aly et al. [198] proposed a method for generating multimodal robotic behaviours based on the

extroversion-introversion personality trait of the human speaker. The system presented is composed of the following modules: a text to speech module, a personality trait recognition module, a natural language generation (NLG) module, a toolkit for translating the generated text into beat gestures, and a metaphoric gesture generator.

The transcription of the speech is sent to the personality recognizer, which performs a psycholinguistic analysis of the text in order to obtain the personality trait of the user (introverted or extroverted). This trait, along with the communicative goal and other parameters, is sent to the NLG module. The generated utterance is passed to the BEAT toolkit, which generates a synchronized set of gestures based on the contextual and linguistic information, as well as on the vocal intonation. Discourse annotations are used by the behaviour generation module to suggest all possible gestures, which are then filtered in order to obtain the most appropriate set of expressions. At the same time, relevant contextual information will be used for generating relevant non-verbal expressions. The output of the BEAT toolkit is an animation script for the verbal and non-verbal behaviour, synchronized based on the duration of the speech. In parallel, metaphoric gestures are generated using Coupled Hidden Markov Models (CHMM), and their amplitude and duration will be adapted to the user's prosodic cues. The metaphoric gestures and the animation script generated by the BEAT toolkit are sent to the robot's behaviour controller, which first models the gestures generated by BEAT according to the user's personality, and then resolves any conflicts that might arise between the different types of gestures generated. Using information from the speaker in the gesture generation process is an approach that has been followed by other researchers. In 2019, Ginosar et al. [199] presented a method for speech-to-motion translation, based on the relation between the monologue speech of a speaker and his/her conversational gestures. The system receives an audio clip of an utterance and generates a sequence of arms and hand motions that match the speaker's gesticulation style. Then the model synthesizes a video of the speaker uttering the speech and performing the gestures. To ensure the smoothness of the generated gestures, the system learns from an audio encoding generated from the entire utterance and predicts poses for the whole sentence at once.

The convolutional network presented in [199] is composed by an audio encoder and a 1D UNet translation architecture. First, a 2D log-mel spectrogram is extracted from the audio clip of the utterance. Then, the spectrogram is sent to the encoder, which downsamples it to a 1D signal through a convolutional process. Next, the 1D signal is received by the UNet translation architecture, and mapped to a temporal stack of pose vectors via an L_1 regression loss. The bottleneck layer of the network gives the system both past and future temporal context, while the skip connections allows for a prediction of fast motion. An adversarial discriminator is used to evaluate if the generated

motions match the style of the speaker. The method for generating co-speech gestures proposed by Ahuja et al. [200] in 2020 also aimed at reflecting the gestural style of different speakers, while maintaining the content of the gesture. The style of the speaker is encoded in a 2D space, and defined by the idiosyncrasy of each speaker and a series of contextual circumstances, like the orientation of the body, or the posture (standing versus sitting, for example).

The proposed model receives a sequence of audio frames and the gestural style of a given speaker and generates a sequence of 2D poses. During training, an audio encoder processes the input speech, while the sequence of poses for the speaker is divided into style and content, and each aspect is managed by a separate encoder. Then, the style of the pose can be either be concatenated with the encoding of the audio or the pose content. A generator with multiple sub-generators is conditioned on the input speech and the speaker's pose for the decoding the output postures. Each sub-generator learns a different mode of the gesture space, which are tied both to style embeddings and audio content. To avoid an overly smooth generation, the poses are fed to an adversarial discriminator, which tries to discern if the pose is real or generated. The generator is then trained to fool the discriminator by decoding realistic poses. Both the content and style encoders, as well as all the sub-generators and the discriminator, are modelled using Temporal Convolution Networks, while the generator is modelled as a 1D version of U-Net. Also in 2020, Yoon et al. [113] presented a new model for gesture generation that used a multimodal context formed by audio, text, and the identity of the speaker. The authors also added an adversarial mechanism for training which, along with the multimodal context, endows the system with the ability to generate human-like gestures that match the content and rhythm of the speech.

An individual encoder is used for each of the three modalities considered. This requires adding padding tokens to the transcription of the speech so it has the same number of time-steps than the other modalities. Both the prosodic features and the transcription are encoded into 32-D feature vectors, while the identity of the speaker is represented by one-hot vectors with all elements set to zero but one, and then mapped into an 8-D style embedding space. These feature vectors are concatenated into a single vector for each time-step, and then passed to the gesture generator (modelled as a multilayered bi-directional GRU network), which outputs a sequence of poses for 10 upper body joints for the next time-step. If the speech is too long, then it is split into 2-second chunks. In order to ensure that the transition between motions is smooth, the system appends the last four motion frames to the feature vector for the first four frames of the current motion. These are considered the seed poses.

Besides considering the identity and the gestural style of speakers, other approaches have relied on other external factors for selecting/generating the most appropriate non-verbal actions. For example, in 2018, Lugrin et al. [201] presented a hybrid approach to predicting cultural-dependant, non-verbal behaviours based on the content of the dialogue. This solution combines theory-based techniques that use cultural theories and theoretical knowledge to build the computational model, and data-driven techniques that rely on multimodal cultural recordings to learn the parameters for the model.

The proposed network is divided in two parts: influencing factors (cultural background and conversational verbal behaviour), and resulting behaviours (semantic content and communicative function of the speech). Based on the influencing factors, a Bayesian network computes the settings for non-verbal behaviours that have to be performed. This work focuses on upper body gestures, and takes into account the type of expression (excluding emblem gestures for being highly culture-specific), the posture for the arms, and the dynamic variation of the expression. Each influencing factor affects to particular parameters of the resulting behaviours. For example, culture will affect the dynamics of the behaviour based on cultural dimensions, while the topic of the dialogue will affect both the gesture and posture types. In order to augment the model using an automated learning process, the authors annotated a series of videos in a dialogue dataset with behavioural attributes. Two datasets were created, one with aligned gestures' dynamics and speech acts, and another without the alignment. The former was used to learn the joint probability distributions of poses and gesture types based on verbal behaviour and culture, while the latter was used for learning the probabilities of gestures' dynamics, based only on culture. A year later, Ghosh et al. [202] proposed an end-to-end system for mapping non-verbal behaviours to the speech of the robot, while adapting the prosodic features of the speech and controlling the engagement of the audience. The model receives the transcription of the speech and selects the most appropriate gesture from a library. At the same time, a TTS module generates the speech of the robot, modulated based on visual feedback about the state of the audience. Both the speech and gesture are fed to the robot's controller, which performs them.

The model was trained with video fragments extracted from TED talks. For each frame of the video, the position of the head and neck of the speaker was obtained using a Convolutional Pose Machine. Then, the frames are modified so the positions extracted for each frame are aligned. Using the same convolutional network, the joint coordinates for both arms are extracted, and converted into relative distance vectors from the neck, which are then clustered. The number of clusters represents the number of dominant gestures, plus a cluster containing residual motions. The library

of gestures used in this work was generated using the centroid of each cluster as reference. The proposed model for selecting gestures is based on Random Forest, and returns the name of the gesture to perform. Regarding the production of speech, the gaze vectors for the people in the audience are used to determine their level of attention. Depending on the variation of this level, the pitch of the robot's voice is either raised, in order to attract attention, or lowered, if the level is high enough. More recently, Xiao et al. [203] presented in 2020 an autonomous system that endows a socially-assistive robot with the ability to learn the correlation between speech and behaviours from context-appropriate human-human interaction examples. The proposed system is able to capture metaphoric and iconic gestures, and generates behaviours that are tied to the context of the application in which the robot is going to be used.

The architecture is divided in four stages: data collection, data processing, text-behaviour mapping, and robot implementation. The data collection step is required to prepare a dataset of human co-speech behaviours that will be used to train the model. The videos used are of either a speaker in front of an audience, or multiple speakers interacting among them, while the behaviours are represented by sequences of poses at given time points. A filtering stage corrects any error, converts the poses from xyz coordinates to joint angles, and filters unfeasible and unrealistic poses.

The transcription of the speech is fed as an input to the machine learning-based mapping model, and was segmented using the punctuation signs in the sentences. The model first compares the joint pose frames in the dataset in order to obtain the dissimilarities between a pair of behaviours. A regression network receives pairs of sentences from the training dataset and the behaviour dissimilarity scores as labels, and seeks to achieve a match between them. When an input text is received, the model starts by finding the sentence from the dataset that shows the lowest dissimilarity, and then selects the motion accompanying said sentence as the behaviour for the input text.

Among the works reviewed in this section, the method proposed by [199] introduced the use of an adversarial discriminator for controlling that the generated expressions match the style of the speaker. This strategy was also followed by other researchers. In 2019, Ahuja et al. [204] introduced a model that uses a joint multimodal space combining language and pose to generate animations based on the input sentence. The system learns the joint space through a curriculum learning approach that prioritizes short and simple sentences over long and complicated ones.

The goal of the proposed system is to generate motions based on a written description (for example, *"a person moves laterally"*). First, a sentence encoder (modelled as a LSTM network) maps

the sentence into a latent representation, while a pose encoder (modelled as a GRU network) does the same for the sequence of poses related to that sentence. A joint translation loss and an end-to-end curriculum-based training ensure that the representations of language and motions are close in the embedding space. The joint translation loss is computed as a sum of both the cross-modal loss between the original sentence and the predicted movement, and the uni-modal translation loss between the original and the reconstructed movements. During training, the system is first optimized to predict two time-steps conditioned on the complete sentence, which leads to the model learning how to predict short pose sequences. When the loss on the validation set starts to increase, the amount of time-steps is increased until the system reaches the maximum number of prediction time-steps. A year later, Ferstl et al. [205] proposed the use of a generative adversarial training paradigm to solve the problem of speech-gesture mapping. The authors also proposed a model for classifying the different phases of a gesture, which will be used to segment the generated motions so the different phases can be judged by the discriminator.

Given an input sequence, the classifier assigns one of six phase labels to each frame of the sequence. The classifier is modelled as a two-layer recurrent network with an additional densely connected layer. Both recurrent layers are modelled as Long-Short Term Memory networks, with the first one being unidirectional, and the second one being bidirectional. Three different models were trained in the presented work, each one using a different number of phases.

The core of the proposed system is the gesture generator. It receives as inputs the pitch of the voice and a vector of 26 MFCC features, and infers a set of 3D joint poses. The inputs are processed by a densely connected network layer, followed by a dropout layer and batch normalization. Then a GRU network generates the joint positions. During the adversarial training of the system, the output of the generator is passed through four separate discriminators. The first controls that the output of the generator follows a realistic phase structure. The second one ensures that the generated joint poses fit the structure of a humanoid skeleton. The third one checks that the outputs of the generator are not repetitive throughout the data batch, even if they are novel patterns. Finally, the last one penalizes motions that are either very slow or very fast, and reduces jitter in the motions, resulting in smoother outputs. The output of all the discriminators is used to compute an average error. Then, a training step of objective numerical errors is performed. Also in 2020, Yu et al. [206] proposed a speech-driven generation method that maps a representation of speech audio to a set of appropriate gestures using a Generative Adversarial Network architecture. It has the advantage of being able to generate multiple gesture patterns for a single speech input using random noises.

In this approach, the model trains a pair of adversarial networks, a generator and a discriminator. The former predicts outputs based on the knowledge learned from the training data, while the latter labels samples as real or generated by the system. The generation network follows an encoder-decoder architecture. The encoder receives the audio for the robot's speech and generates a 256-dimensional representation. The decoder receives the output of the encoder combined with 10-dimensional random noise to add variability to the motions generated, and outputs a sequence of 3D poses mapping with the speech audio. In order to transfer the generated motion into the robot, the 3D position of each joint has to be transformed into the necessary joint rotation angles for the robot, taking into account the constraints that the robotic platform might introduce. The discriminator starts by performing the same 1D convolution of the audio clips that the encoder performs. In parallel, either the real gesture or the output of the decoder is also encoded. Both the audio and gesture encodings are concatenated and then fed into a one layer GRU network. Finally, the final hidden state of the network is sent into a last Multilayer Perceptron that has to distinguish if the gesture is real or not (if it matches the input speech audio).

5.2.3 Comparison between approaches for co-speech gesture prediction/generation

Here, a comparison between all the works reviewed above will be conducted, based on a series of characteristics that better represent the differences between the approaches, and that are relevant for the application considered in this thesis: the development of lively robots for Human-Robot Interaction. Three features have been selected for the comparison between approaches. The result of this comparison can be observed in Table 5.1 The first two were extracted from the review conducted in section 4.2, although their definition has been adapted to the particular characteristics of this study. The first one is *Multimodality*, and in this chapter, it is understood as the different sources of information that can be used to generate the expressions (prosodic features of the speech, information about the other peer, contextual factors, etc...). The outputs of the models are not considered in the analysis, as all methods reviewed in this section generate either sequences of poses (the majority of approaches) or the name of the gesture/gestural category that has to be attached to the speech.

The second feature, *gesture design*, is related to how the expressions will be modelled. This is a key task when defining a expressiveness model for a communicative agent, both virtual and physical. From a general point of view, two approaches can be followed: either the behaviours are designed

beforehand and stored in a library, or the model learns to generate appropriate expressions. The former has the advantage of allowing the developers to design expressions that perfectly convey the intended message, combining all the available communication channels. On the other hand, relying on a finite library of gestures constrains the actions that can be performed. Automatic generation of behaviours based on the speech leads to systems with a bigger range of actions, although the gestures performed can be perceived as more generic.

Finally, the last feature used in the comparison is the algorithms used to create the co-speech gesture generation/prediction models. In co-speech gesture generation, the task of the model is to learn the mapping between the input information and the appropriate non-verbal behaviours that will accompany the verbal communication. The inputs to the model and the outputs that have to be generated will play a role on how the architecture of the proposed model has to be designed.

5.2.3.1 Multimodality

Regarding the inputs to the model, a higher variability can be observed between works. Because non-verbal behaviours are designed to complement verbal communication, the generated speech of the agent has to be always one of the inputs to the model. Here, a distinction can be made depending on the type of information that the system extracts from the speech. Authors like Aly et al. [198], Yoon et al. [195], or Xiao et al. [203] based their approaches on the transcript of the speech, while others like Hasegawa et al. [193], Kucherenko et al. [194], or Ferstl et al. [205] relied on audio features. One of the most used features are the Mel-frequency cepstral coefficients, or MFCCs, which represent the short-term power spectrum of the audio signal. Kucherenko et al . [194] fed the MFCCs to their baseline system, although they also considered using alternative features. Ferstl et al. [205] combined the MFCCs with the pitch of the audio signal. Other approaches include the use of prosodic features, as shown by the work of Chiu et al. [190], or directly encoding the audio signal. This last method appears in the approaches presented by Yu et al. [206] or Ahuja et al. [200]. Some works combine both audio features and text to improve the obtained results. Chiu et al. [191] used the prosodic features of the speech alongside the content and part-of-speech class for each word in the text. Another example is the work of Kucherenko et al. [112], where the BERT encoding of the speech transcription and temporal information about how the sentence is uttered is combined with log-power mel-spectogram features extracted from the audio signal. Pérez-mayos et al. [196] decided to use the prosody of the speech to select beat gestures, while text is used for the remaining categories.

Reference	Algorithm used	Gesture design	Multimodality
[190]	CRF (annotation), GPLVM (generation)	Input labelled, poses selected for labels and interpolated	Prosodic features
[191]	Neural networks and 2^{nd} order linear chain temporal contingency	Gestures describing the shape of a motion assigned to the input	Linguistic (words and POS) and prosodic features
[192]	Probabilities $\rightarrow$ function $\rightarrow$ motion. Beat gestures tied to prosody peaks	Generated based on prosody (beats) and content (other). Speaker-dependent	Text, prosody, dialogue act information
[193]	Bi-directional LSTM	Poses predicted + temporal filtering	Audio features (MFCC)
[194]	Denoising Autoencoder	Encoder codifies the speech features, decoder reconstructs the motions	Audio features (MFCC)
[112]	Feed-forward network plus several fully-connected layers	Sequence of poses, conditioned on previous poses	Text (BERT encoding, and duration) and audio (log-power mel-spectogram features)
[195]	GRU (encoder, decoder)	Encoder codifies speech features, decoder reconstructs the motions	Text
[196]	PoS-based: gestures tied to keywords. Prosody-based: gestures tied to pitch peaks	Handcrafted gestures	Text, prosody (pitch values)
[197]	Dynamic Bayesian Network constrained on discourse function or target gesture	Gestures constrained by discourse function or expression type	Prosodic features

Table 5.1: Comparison among the works presented in the state of the art review. Each approach has been evaluated based on the algorithm used, the design of the expressions, and the multimodality considered.

Reference	Algorithm used	Gesture design	Multimodality
[198]	Rule-based gesture generation. CHMM (metaphoric gestures)	Generated from the text, based on the user's personality	Text
[199]	CNN, Adversarial discriminator	Generated based on speaker style	Audio (2D log-mel spectogram)
[200]	TCN (encoder & discriminators), 1D Unet (generators, 1 per style)	Sequence of 3D poses, constrained on style selected	Audio signal and style
[113]	Generator: Bi-directional GRU, Discriminator: multilayer GRU	Sequence of 3D poses, constrained on user ID	Text, audio (encoded waveform), speaker ID
[201]	Bayesian network + influencing factors. Augmented through learning	Gesture & pose type based on speech act and topic. Altered based on culture traits	Culture factors, topic, speech act
[202]	Random Forest	Generated based on clusters in design space	Text
[203]	Regression network (text to expression). Fast Dynamic Time Warping (expression diffs.)	Handcrafted. Expression-dependent delay for synchronization	Text (GloVe word embeddings)
[204]	LSTM (speech encoder), GRU (pose encoder, decoder)	Generated based on natural language descriptions of motions	Text
[205]	LSTM (phase classification), GRU (motion generation). Four discriminators	Generated based on speech features. Discriminators control generation during training	Audio features (MFCC + F0)

Table 5.1: Comparison among the works presented in the state of the art review. Each approach has been evaluated based on the algorithm used, the design of the expressions, and the multimodality considered.

Reference	Algorithm used	Gesture design	Multimodality
[206]	Encoder: 1D CNN + GRU, Decoder: GRU + MLP. Discriminator: 1D CNN + GRU + MLP	Sequence of 3D poses	Audio signal

Table 5.1: Comparison among the works presented in the state of the art review. Each approach has been evaluated based on the algorithm used, the design of the expressions, and the multimodality considered.

Besides the use of text, audio, or both to represent the content of the speech, multiple authors decided to constrain the gestures generated by their models according to different external factors. For example, both Ahuja et al. [200] and Ginosar et al. [199] presented methods for generating behaviours that represent the style of the speaker. While the model proposed by Ginosar et al. can only be trained to convey the style of a particular speaker, Ahuja et al. used a multi-generator structure that allows the system to receive the desired style as an input, and use it during the generation process. The work presented by Yoon et al. [113] introduces a similar idea, where the identity of the speaker is passed to the model in order to generate appropriate motions. Beyond the style of the expressions, other authors have experimented with the use of discourse functions in order to create non-verbal behaviour that better reflect the communicative intention of the speech. For example, Ishi et al. [192] conducted an study to understand the relationship between non-verbal behaviours and dialogue acts, and as a result used the dialogue acts identified in the speech to filter the gestures generated by their system. Sadoughi et al. [197] included the discourse function of the speech (for example, if it is a question) as a constraint node in their Bayesian network. In this work, the authors also presented a model that used the gesture type instead of the discourse function. In the work presented by Ferstl et al. [205], the authors focused on enforcing that the gestures generated presented a realistic phase structure (the different motion phases in which they divide the gestures), using a phase classifier to extract said structure from the generated expression and an discriminator to reject all gestures that are not deemed realistic. Finally, Lugrin et al. [201] presented a model for generating culture-dependent behaviours for virtual agents. While the content of the speech (topic and speech act) play a role on selecting the posture type and gesture type, the cultural background of the speaker constrains that selection process, and at the same time defines the dynamic aspects of the behaviour.

5.2.3.2 Gesture design

Among the authors that opted for relying on predefined expressions (this can vary from handcrafted poses to gestures including the complete motion trajectories), several approaches can be observed. In the works presented by Chiu et al. [190, 191], the speech is labelled with the name of the gestures that have to accompany it. In their earlier work (2014), the choice was to either use a beat gesture or not. Based on this, the appropriate motion poses were selected from the motion space, and the trajectories between poses were interpolated. In their second work, the labels indicate the shape of the gesture (head nod, rest pose, etc...). Ghosh et al. [202] presented a method that extracted non-verbal behaviours from training videos, and then clustered them to generate the available expressions. Their model then predicts which one of the clusters obtained should be selected based on the speech, and a gesture is instantiated based on the position of said cluster's

centroid. Authors like Pérez-Mayos et al. [196] or Xiao et al. [203] handcrafted their gestures using the tools provided by the robotic platforms used in their works (in both cases, a Nao robot). Pérez-Mayos's work synchronized these predefined expressions by associating their triggering to either keywords or pitch peak patterns in the utterance. On the other hand, in Xiao's work, a single expression is assigned to the speech, and synchronization is achieved through the introduction of a small delay in the speech that allows the robot to reach the required position to perform the expression.

The remaining works analysed in this section opted for an automatic generation of non-verbal behaviours. Among these approaches, the majority of the authors have designed models that learn to predict sequences of joint poses, either 3D or 2D, that can then be mapped directly into the agent. Authors like Aly et al. [198] opted for using rule-based generation approaches. The work of Ishi et al. [192] could be also considered as a rule-based system, as non-beat gestures are selected based on the presence of specific keywords in the speech, while beat gestures are parametrized according to the pitch of the utterance. The remaining authors designed data-driven models that learn which pose should be generated based on the inputs received. Here, two works have to be highlighted. The approach presented by Lugrin et al. [201] combines a data-driven approach for gesture generation with a theory-based approach to constrain said behaviours to the cultural background of the speaker. On the other hand, the work presented by Ahuja et al. [204] is unique among the ones reviewed, as the gestures are not designed to accompany the text introduced to the system, but instead this text represents a natural language description of the motion that has to be generated.

5.2.3.3 Algorithms used

Automatic generation of expressions usually relies on concepts and structures extracted from the field of machine learning, specially with the advances that deep learning techniques have been experiencing nowadays. Among these, a common architecture for this task is to treat the generation of expressions as a sequence-to-sequence problem, where given an encoded input sequence, the system decodes an appropriate output sequence. Authors like Ahuja et al. [204], Yoon et al. [195], or Yu et al. [206] followed this approach. Recurrent Neural Networks tend to be used to model the encoders and decoders in this systems, due to their ability to display temporal dynamic behaviour. Some authors, like Yoon et al. [113] or Hasegawa et al. [193] used bi-directional recurrent networks to model their generators, a Long-Short Term Memory neural network in the first case, and a Gated Recurrent Unit network in the second. Lungrin et al. [201] and Sadoughi et al. [197] decided to model their systems using bayesian networks, where the external constraints considered by each author are included as nodes of the network. Regarding the training of the model, adversarial

strategies are fairly common in the works reviewed in this section. Examples of these strategies can be observed in the works of Ferstl et al. [205], Ginosar et al. [199], Yoon et al. [113], or Yu et al. [206].

In their work presented in 2014, Chiu et al. [190] used a combination of Conditional Random Field for gesture annotation and Gaussian Process Latent Variable Models for generating the appropriate motions based on said annotations. A year later, they presented a new version [191] that combined the advantages that deep neural networks present for mapping complex relations and second-order linear chain for modelling the temporal relationship between speech and gestures. Aly et al. [198] proposed a combination of Coupled Hidden Markov Modules for generating metaphorical gestures, and a rule-based generation process for the remaining categories. In the work of Ghosh et al. [202], gesture prediction is performed using Random Forest. Ishi et al. [192] decided to use conditional probabilities to represent the connections between speech content and gesture functions, and between functions and motions. Pérez-Mayos et al. [196] combined a part-of-speech based approach, where gestures were associated with keywords in the speech, while beat gestures were generated based on the prosodic content (in particular, the pitch graph). Finally, the work presented by Xiao et al. [203] focuses on finding the correlation between the dissimilarities between two sentences and the dissimilarities between their associated behaviours. This correlation is then used to find in the dataset the sentence that is the most similar to the speech that has to be enhanced with non-verbal behaviour. The gestures associated to the selected utterance will be the ones attached to the speech.

5.2.4 Comparison with the solution proposed in this thesis

Overall, the work presented in this thesis models the problem of co-speech gesture generation as a prediction task, where the model has to select the most appropriate gestures from a library based on the speech that has to be uttered, and also the robot's communicative intention. This perspective can be observed in the works of Chiu et al. (both in the work they presented in 2014 [190] and the one from 2015 [191]) or Ghosh et al. [202]. The approach proposed in this thesis follows an approach that is similar to the one presented by Chiu et al. in 2015, which combines the advantages of deep neural networks with the temporal modelling capabilities of Conditional Random Fields in order to predict the gestural label that has to be assigned to each word of the speech. The model presented in this dissertation in turn uses LSTM networks to encode the sequence of words, part-of-speech label, and communicative intentions, and then uses Conditional Random Field to predict the sequence of gesture labels. The two main differences with the model presented in this thesis are the inputs to the model and the approach followed for identifying the list of gestural signs. While Chiu et al.

combine text and prosody as inputs to the model, the solution developed in this dissertation relies exclusively on the text, although it adds the communicative goal of the sentence to the words and their part-of-speech classification. Thus, this thesis tackles the problem as a two-step process, where first the model determines the communicative intention of the speech (request personal information, ask a general question, show enthusiasm, etc...), and then uses this to generate the sequence of multimodal gestures. Regarding the gestural signs, this thesis proposes to use labels that represent the communicative meaning that the gesture has to convey (apologize, greet, agree, emphasize, etc...), instead on describing the shape of the gesture. Then, a rule based approach is used to synchronize the gestures connected to those labels and the speech. This is another difference with the work of Chiu et al., in this case, the paper presented in 2014 [190], where the gesture annotations generated by the Conditional Random Field were used to select motions from the motion space, and then the complete trajectory was interpolated.

Other approaches have used the discourse function as a constraint for gesture generation. For example, Ishi et al. used dialogue acts to filter if a generated expression should be executed. Sadoughi et al. [197] included the discourse function as a constraint in their Bayesian network, while Lugrin et al. [201] follow a similar approach and use a speech act node to affect the selection of both gesture type and posture. The difference between the method developed in this thesis and both approaches is the discourse functions selected. While both Sadoughi et al. and Lugrin et al. use speech acts exclusively related to the function of the utterance (information request, statement, answer, etc...), the communicative intentions considered in this work, while still describing the utterance's function, also take into account the content of the message conveyed (for example, information requests can be either aimed at retrieving generic information, information about the other speaker, or an opinion). Another important distinction is that the information about the communicative intention of the utterance is not fed to the model, but predicted by it. Regarding the selection of the appropriate gesture from the dataset, this dissertation proposes the use of a set of rules tied to the gesture labels for both gesture selection and synchronization. A similar approach was proposed by Ishi et al. [192], where gesture functions were predicted for the speech, and then motions were associated to those functions. In their approach, gesture motions from their dataset were clustered based on similarity. Then, depending on the conditional probabilities between the predicted gesture functions and these clusters, one of them is selected, and a random motion is extracted. In this dissertation, the mapping is simpler, as each gesture label has an associated list of predefined multimodal behaviours, and the proper one can be extracted based on the length of the utterance segment tagged with said gesture label. Finally, regarding the design of the gestures, the approach proposed in this thesis can be compared to the works of Pérez-mayos et al. [196] and Xiao et al. [203], where gestures are

handcrafted using design tools provided by the robotic platform. The main difference is that the approach proposed in this dissertation takes the list of expressions that have to be performed and the utterance, and combines them into a single multimodal expression that defines the relationships between gestures and the speech.

Finally, the model for predicting and synchronizing co-speech gestures proposed in this thesis has been integrated in the HRI architecture of our social robot, alongside with a new Liveliness module that generates random actions for each of the robot's interfaces. Said actions can be modulated based on the internal state of the robot (in the current version, only the affect states are considered). The model presented in this chapter will work alongside the new Liveliness model, using one or the other based on the presence of the robot's speech.

5.3 Strategies developed for endowing social robots with a liveliness appearance

The work presented in this chapter aims at endowing a robot with a lively appearance through the use of behaviours that have no particular communicative goal, and instead are performed with the sole purpose of increasing the robot's animacy. The proposed approach to the expression of animacy has been divided in this dissertation in two different methods: (i) the pulse-based liveliness module uses a signal that resembles the heartbeat of a human to generate random expressions through each communicative channel at a certain rate; and (ii) the co-speech gesture prediction method uses a predefined library of expressions and selects the most appropriate sequence depending on the content of the robot's speech.

Both methods have been integrated in the robot's software architecture and are able to work together. During the periods of time where the robot is awake, the pulse-based liveliness module will be continuously generating actions (modulated depending on the robot's heartbeat, which in turn depends on the robot's internal state), while the co-speech prediction method will only be used when one of the CAs in the HRI Manager sends an expression to the Expression Manager that only contains a utterance. In order to solve any potential conflicts between these two sources of expressions (the Liveliness module and the HRI Manger), actions generated by the pulse-based method will have the lowest priority, as they do not transmit communicative information, and thus are considered to be less vital during an interaction. When the Expression Manager has to execute an expression, part of the information that is retrieved during this process is the list of

communicative interfaces that the expression needs to use. Whenever an action coming from the pulse-based liveliness is received, the Expression Manager checks if the required interface (the actions generated by the Liveliness module are always unimodal) is free. If it is, then the action is performed. Otherwise, it is discarded. The pulse-based liveliness module will be presented in Section 5.3.1.

If the expression requested by the HRI Manager involves exclusively uttering a sentence, then the Expression Manager passes this sentence through the co-speech gesture prediction process, and then continues with the execution process described in 4.5.3. The detailed operation of the co-speech gesture prediction process will be described in Section 5.3.2.

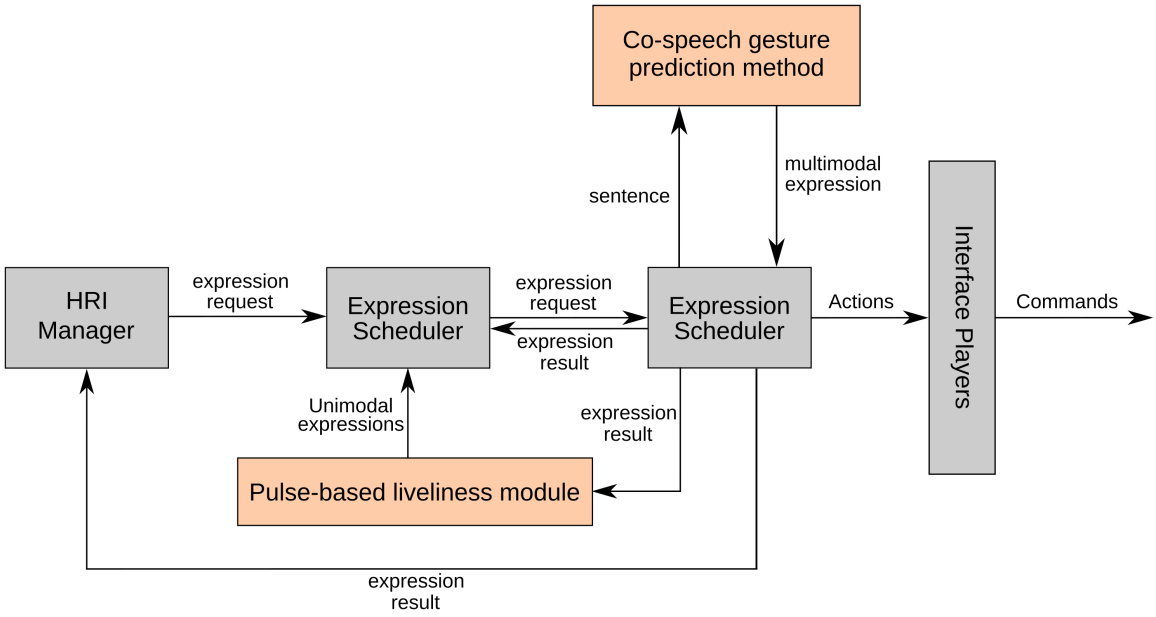


Figure 5.2: Integration of the proposed liveliness approaches in the HRI System introduced in Chapter 2. The light orange boxes represent the two liveliness modules.

5.3.1 Pulse-based liveliness

The pulse-based liveliness method, shown in Figure 5.3, creates unimodal actions for each communicative interface based on a signal that represents the heartbeat of the robot. Whenever the internal state of the robot variates, the module updates this signal, which in turn translates in a modulation of the actions generated. Currently, the only internal state considered is the robot's affect state. During the execution of the pulse-based liveliness module, two main tasks can be identified: (i) update the control parameters that depend on the robot's state and that will be used during the

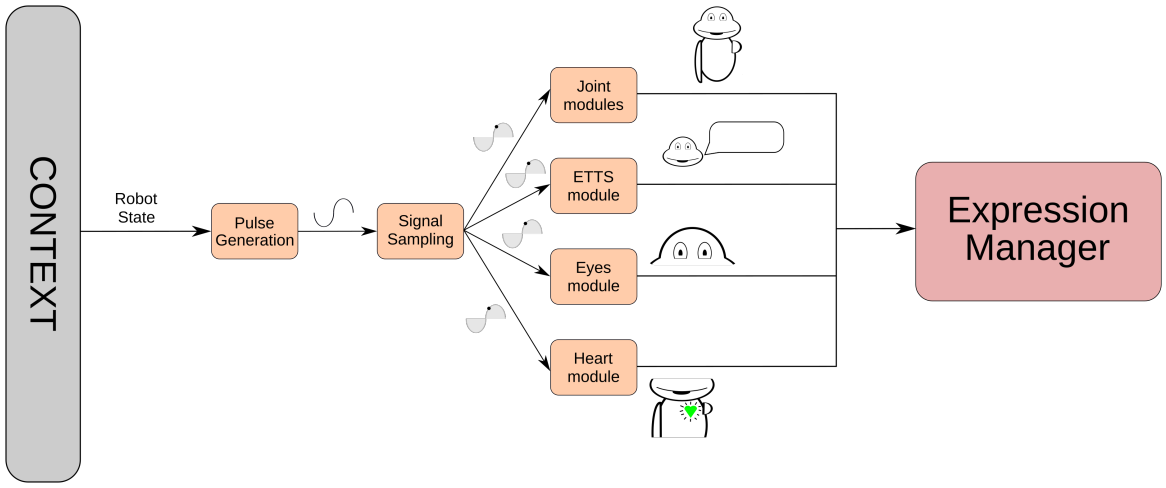


Figure 5.3: Schematic of the pulse-based liveliness approach and its integration in the robot's architecture.

creation of the actions; and (ii) create unimodal expressions for each of the available interfaces. The pulse-based liveliness module is divided in two main components: (i) the Signal module is in charge of generating the signal that will be used to shape the behaviours created; and (ii) the Interface modules receive samples extracted from the robot's pulse, and use them to create actions for each of the robot's interfaces (there is an individual Interface module for each communicative channel).

5.3.1.1 Generation of the robot's heartbeat

It has been argued in several sections of this manuscript that one strategy that helps to create more natural interactions between the robot and the user is the proper modulation of the robot's expressiveness, so it can be adapted to different situations. This implies, for example, conveying a variety of affect states with the generated behaviours. Similar to one of the modulation methods presented in Section 4.5.4, the pulse-based liveliness module achieves this through the use of a signal, the so-called robot's pulse, that will define how each interface behaves. This pulse can be modified to reflect each of the robot's affective states. In the proposed architecture, this process is controlled by the Signal module.

This module generates a sinusoidal signal that represents the heartbeat of the robot. The Signal module samples the heartbeat at a fixed frequency (10 Hz) in order to obtain the values for three pulse parameters: (i) the value of the signal at that particular point in time; and (ii) the frequency and (iii) amplitude of the signal. Every time the signal has to be sampled, the Signal module extracts the state of

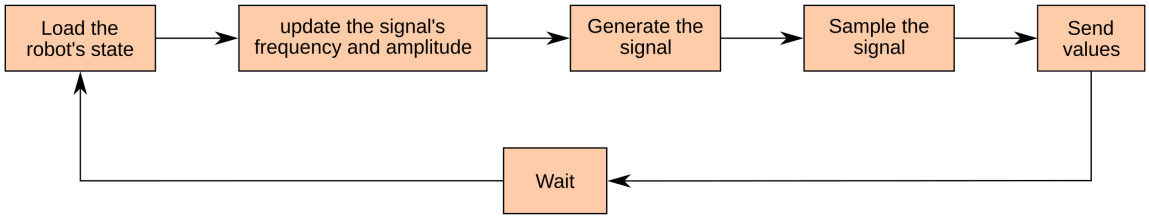


Figure 5.4: Activity diagram that shows the process followed for generating the robot's pulse.

the robot from the context. This state is defined by a string holding the name of the state (for example, happy, sad, angry...). In case the robot's state has changed since the last sampling cycle, the Signal Module retrieves the pulse's amplitude and frequency associated to the new state. These can have values ranging from 0 to 100, and have been empirically determined for every possible state considered. The robot's pulse will always have a frequency and amplitude bigger than 0 while the robot is in operation, ensuring that the pulse-based liveliness module will be generating behaviours continuously. When the robot is asleep, the parameters of the signal descend to almost 0, and thus the robot does not perform any liveliness behaviour. The steps followed by the signal module are shown in Figure 5.4.

5.3.1.2 Generation of actions based on the robot's pulse

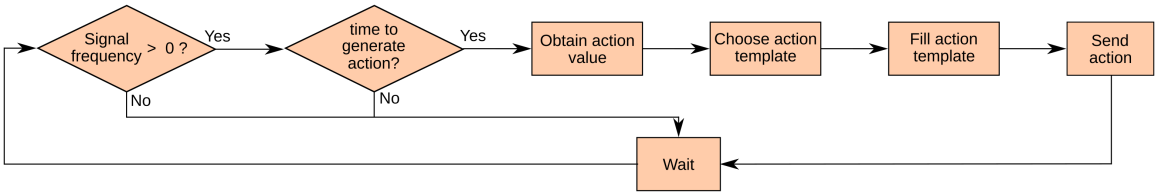


Figure 5.5: Activity diagram that shows the process followed by the Interface modules to create actions.

While the Signal module controls the update process of the signal parameter's values, the Interface modules use these values to generate the unimodal actions that will be used to enhance the animacy of the robot. An example of how the signal parameters affect the robot's actions can be seen in Figure 5.6. The Interface modules are robot-specific, as they depend on the available communication channels and the configuration parameters they need (for example, while one robot's arms might have a single joint in the shoulder, another robot can have arms that resemble the joint configuration of a human arm). The Interface modules perform three tasks: (i) generate a new action, (ii) control the frequency with which actions will be generated, and (iii) check if new values have been

received for the control parameters. While the first two are done sequentially, the last one is done asynchronously. The module updates the values used for generating actions whenever new ones are received from the Signal module. These three tasks will be repeated indefinitely, as shown in Figure 5.5.

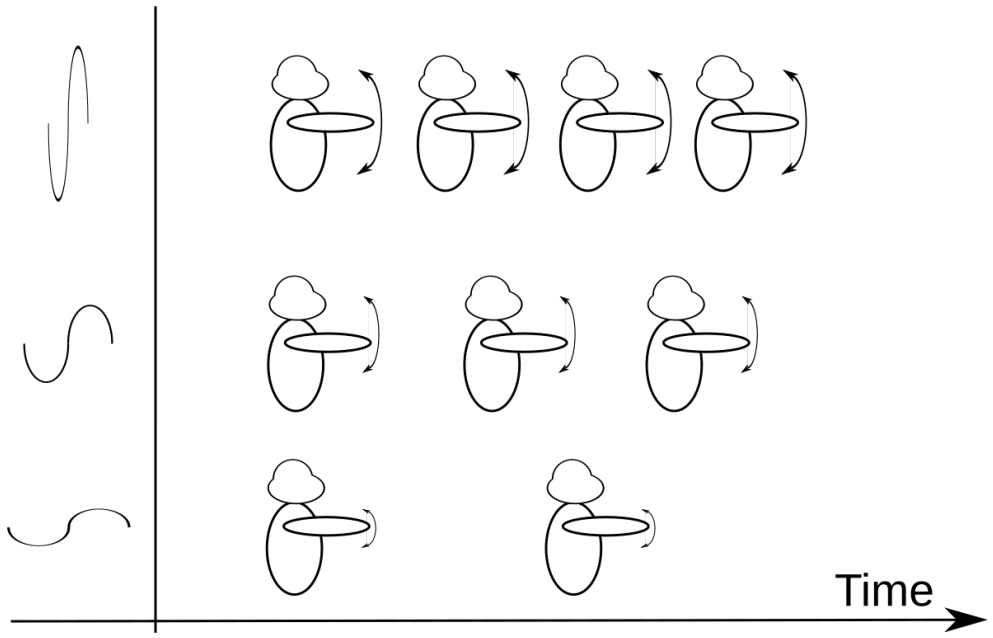


Figure 5.6: Example of the effect that the signal parameters have over the actions generated by the pulse-based liveliness module.

In the current implementation of the pulse-based liveliness, the following Interface modules exist:

- **Joint modules:** motions are generated using templates that describe the general trajectory of the limb (for example, one template can indicate a motion where the robot turns to the left and goes back to looking forward, while a second template can define a motion where the robot first turns left, then right, and then goes back to looking forward). During the action generation step, the module selects randomly one of the available templates, and defines the trajectory points based on the values of the signal parameters. While individual modules exist for each joint, both arms are controlled by the same module, as the actions created can involve motions of either arm or both.
- **ETTS module:** generates non-verbal sounds to be uttered by the robot at random intervals. The module maintains a list of available sounds that can be used, and extracts one of them randomly whenever an action has to be generated. Opposite to the other modalities, the

speech-based liveliness is not affected by the sampling of the heartbeat signal, only by its frequency.

- **Eyes module:** changes the direction of the gaze, selecting one of nine possible values (eight positions placed in 45 degree intervals, and a ninth position where the eye is centred). This selection is performed based on the values for the signal parameters. The behaviours created by this module always include two commands, one for performing the selected gaze change, and a second one to return the eyes to the normal position (looking forward).
- **Heart module:** changes both the colour and the fade frequency of the LED that the robot has in the chest, acting as its heart. The colour is completely independent from the values extracted from the heartbeat signal, but is instead tied to the robot's affect state. The fade frequency, on the other hand, is connected to the signal's frequency.

In the software architecture presented in Chapter 2, the requests for executing gestures that the Expression Manager receives come usually from two main sources: the HRI Manager and the pulse-based liveliness module. While the expressions requested by the CAs seek to achieve a given communicative goal, the actions generated by the pulse-based liveliness module do not. Due to this, a decision was made to assign always low priority to the expressions requested by the pulse-based liveliness module, so the expressions required to advance the interaction with the user can always be performed in the event of a conflict between several expression requests. Listing 5.1 shows an example of how an Interface module would use the customizable gesture template (presented in Section 4.5.4.1) to request the execution of an action. In the example presented, the parameter *motion* holds the values required to configure the arm motion generated by the Interface module (the trajectories that they have to follow, as well as the speed and acceleration). These parameters have been obtained based on the robot's pulse. In the request shown in Listing 5.1, the action generated requires the motion of both arms (as shown by the value associated to the key *interfaces_needed*). The motion parameters are stored in a key-value pair array inside the request.

```
void Liveliness_arms::send_msg(common_msgs::KeyValuePairArray motion){

    common_msgs::KeyValuePair kvp_arms;
    interaction_msgs::Expression arms_msg;

    double tem =ros::Time::now().toNSec() * 1e-6;
    std::vector<string> aux;
    aux = this->split(boost::lexical_cast<std::string>(tem), '.');
    this->arms_msg.id = aux[0] + aux[1];

    arms_msg.name="customizable_gesture";
    arms_msg.emitter="liveliness_arms";
    arms_msg.priority=0;

    kvp_arms.key="interfaces_needed";
    kvp_arms.value = "leftArm|rightArm";
    arms_msg.params.push_back(kvp_arms);

    arms_msg.params.push_back(motion);

    this->_pub_arms.publish(arms_msg);
}
```

Listing 5.1: Example of how an Interface module would request the execution of an action using the customizable gesture

5.3.2 Co-speech gesture prediction module

The expressions generated by the pulse-based liveliness module are completely independent from whatever the robot is doing, or from the communicative objectives that the robot's applications might want to achieve. This might lead to unnatural situations during interactions, for example where the robot would be talking to a user in front of it, but continuously moving its body and face away from him/her. According to participants that were invited to test the robot, this was perceived as unnatural, and hindered the quality of interactions. This feedback pointed towards the necessity of developing a way to integrate the display of animacy with the achievement of the robot's communicative goals. Consequently, a new approach for conveying a lively appearance during speech-based interactions was proposed. This new approach focuses exclusively on verbal communication, as this type of messages represent a big portion of the communicative actions performed by the robot, while involving a smaller expressiveness display (as none of the other

interfaces are used). Using this method, application developers can focus on designing interactions between the robot and the user using exclusively verbal communication, and leaving all the other interfaces free. The proposed approach can take advantage of this by selecting the non-verbal expressions that better suit the verbal messages defined by the developers.

The goal of the co-speech gesture prediction module is to enhance the robot's utterances with the addition of non-verbal behaviours that on one hand aim at increasing the animacy of the robot, while on the other try to convey the same communicative goal being transmitted by the voice. There are two strategies that can be followed in order to achieve this goal. The first solution is to generate multimodal actions from scratch based on the contents of the speech message. This approach has the advantage of freeing developers from the need to design the non-verbal expressions manually. But gestures generated automatically can end up being more generic and lacking the level of detail that a handcrafted expression has. This is where the second approach comes into play: using a predefined library of expressions and selecting the most appropriate one depending on the message being conveyed. This solution leads to a few issues that have to be addressed:

- It is necessary to develop a model that can select the appropriate expressions given the speech of the robot. While there are multiple solutions that can be followed to achieve this, the one implemented in this thesis is to treat the problem as a labelling task, where the model has to tag the transcription of the speech with a series of labels that represent which gesture/s should be performed.
- An appropriate label set has to be defined. On one hand, having a limited number of labels can result on a very generic interaction, where the same gestures are used continuously. On the other hand, an excessive number of labels can make relationships between sentences and sequences of labels hard to learn for a model.
- A mapping between the labels generated by the model and the expressions stored in the library has to be implemented.
- Finally, the expressions selected have to be synchronized to the appropriate point of the speech.

5.3.2.1 Model for predicting communicative intentions and gesture semantic values

The co-speech prediction module proposed in this thesis works as follows. The model receives as an input the transcription of the speech, which is first tagged with the Part-of-Speech information. Both the words in the speech and their part-of-speech tags are passed through a first model, which in turn predicts the communicative intention that the speech tries to achieve. The predicted intentions are

then added to the rest of the input information and passed through a second model, which assigns to each word in the speech a label indicating the *semantic* value of the gesture that should accompany it. Finally, the module maps the predicted sequence of semantic values to a list of expressions, which are executed by the Expression Manager following the procedure described in Section 4.5.

While a particular sentence usually only has one intention that do not change at any point of the speech, the same is not true for the semantic values. One sentence can have more than one gesture associated. Thus, dividing the process in two steps (first predicting the intention, then predicting the semantic values) makes it easier for the models to learn the appropriate relationships that will lead to more accurate predictions. The models used for these two steps have an identical structure, differing only in their inputs and outputs. A decision was made to use an hybrid approach that combines a bidirectional Long-Short Term Memory neural network (LSTM network, from now on) and Conditional Random Fields (or CFR) to generate the appropriate label sequence. One of the strengths of LSTM networks is their ability to learn dependencies between all elements in an input sequence. In the case of the proposed model, the input sequence corresponds to the robot's utterance, transformed into a sequence of tokens. CRFs, on the other hand, are able to consider dependencies between labels in the output sequence (in this case, either the sequence of labels indicating the utterance's intention, or the semantic value of the gesture that has to be paired with the speech) based on features of the observations that will be paired with the labels (in this case, the encoding of the utterance generated by the LSTM network). This combination has been used in multiple sequence labelling problems, including Named Entity Recognition [207], Dialogue Act tagging [208], or keyphrase extraction [209].

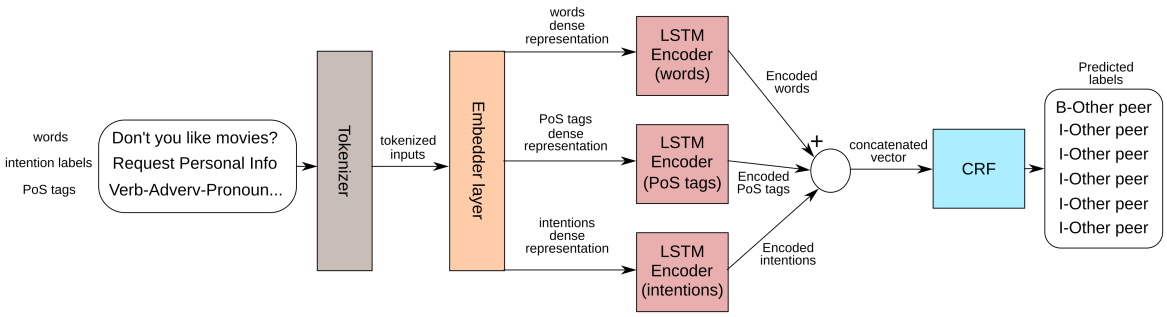


Figure 5.7: Architecture of the model proposed. This example corresponds to the model used for predicting gestures.

Figure 5.7 shows the structure of the model for predicting semantic values for the gestures that should be performed alongside the speech. Both models have been designed using AllenNLP [210].

This is a library designed for applying deep learning methods in Natural Language Processing (NLP) tasks. It has been built over PyTorch, a machine learning library. AllenNLP implements a series of abstractions that help to modularize the process of building a model for NLP, simplifying the design phase. It also allows to define high-level configuration files for training and evaluating models, and provides methods that handle some common problems related to NLP tasks.

5.3.2.2 Pipeline of the prediction models

The inputs to the proposed models are the words in the utterance that has to be labelled, their respective Part-of-Speech labels, and, in the case of the model used for predicting the semantic values of the co-speech gestures, a series of intention labels that indicate the communicative goal that the utterance tries to achieve. Once the inputs have been received, the the following steps are performed:

1. Tokenize the inputs, represent each token with an ID, and send these representations to the model
2. Pass the IDs through an embedder layer in order to obtain dense representations in a vector space
3. Pass the obtained representations through stacked bi-directional LSTM encoders in order to generate a fixed size internal representation. This model uses two recurrent layers, which means that the output of a first LSTM network is passed through a second LSTM network in order to obtain the output. The words, Part-of-Speech tags, and intentions (if present) are encoded separately, using their own encoders.
4. Concatenate the output of each encoder to obtain a single vector, and send it to the linear-chain Conditional Random Field model.
5. Use the Viterbi algorithm [211] to retrieve the more probable sequence of labels (intentions or semantic values) given the received inputs.

ELMo representations have been added to the baseline model for obtaining the dense representations of the tokenized inputs. ELMo [212] (Embeddings from Language Models), uses a bi-directional LSTM model that can generate word representations taking into account the context in which they are used. While traditional word embedding models maintain an embedding matrix where look-up searches are performed to obtain the vector representation for a word, ELMo passes the text through the deep learning model and generates the representation in the moment.

During training, the CRF model uses the negative log-likelihood with the correct set of labels as the loss. The models have been trained using the Adam optimization algorithm [213], a method for optimizing stochastic objective functions. A summary of the design parameters, empirically determined, can be seen in Table 5.2.

Baseline model embedding size	200
Elmo embedding size	1024
Encoder input size	1224
Encoder hidden size	200
Encoder recurrent layers	2
Dropout rate	0.5
Epochs for training	100
Learning rate	0.001
Patience	25
Gradient norm	5.0
Iterator batch size	2

Table 5.2: Configuration parameters used in the design of the proposed pipeline.

5.3.2.3 Generation of the datasets used to train the models

Two different datasets have been developed for training the proposed models. The first dataset, which was used to train the model for predicting the intention/s of a utterance, includes three elements per entry: (i) a sentence; (ii) the list of tokens that are used to represent this utterance; and (iii) the list of communicative intention labels that correspond to each token. The tokens, in turn, include the word, its lemma (the root of the word), and a Part-of-Speech tag. Labels use the IOB format, which uses prefixes to indicate if a label is the beginning (B-) of an entity (a group of identical labels), or if it is inside (I-) an entity. In this particular case, all the tokens in the utterance belong to an entity, so the O- prefix will not be used. The sentences used to create the dataset were extracted from the Cornell Movie Dialogs Corpus [214]. The second dataset, which was used to train the model for predicting co-speech gestures, is a copy of the first one, with the addition of the list of semantic value labels. Figures 5.8 and 5.9 show examples of instances from both datasets (the same instance in both datasets).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<example id="0120">
<sentence>How was your trip? Profitable?</sentence>
<tokens>
<token id="1" lemma="how" pos="ADV" surface="How" />
<token id="2" lemma="be" pos="VERB" surface="was" />
<token id="3" lemma="-PRON-" pos="PRON" surface="your" />
<token id="4" lemma="trip" pos="NOUN" surface="trip" />
<token id="5" lemma="?" pos="PUNCT" surface="?" />
<token id="6" lemma="profitable" pos="ADJ" surface="Profitable" />
<token id="7" lemma="?" pos="PUNCT" surface="?" />
</tokens>
<intentions>
<token id="1" value="B-REQUEST_PERSONAL_INFO" />
<token id="2" value="I-REQUEST_PERSONAL_INFO" />
<token id="3" value="I-REQUEST_PERSONAL_INFO" />
<token id="4" value="I-REQUEST_PERSONAL_INFO" />
<token id="5" value="I-REQUEST_PERSONAL_INFO" />
<token id="6" value="I-REQUEST_PERSONAL_INFO" />
<token id="7" value="I-REQUEST_PERSONAL_INFO" />
</intentions>
</example>
```

Figure 5.8: Example of an instance taken from the dataset developed for the intention prediction model.

The possible communicative intentions were obtained through a manual process, where an annotator reviewed the sentences present in the corpus used to train the model, and proposed the communicative intention (or intentions) that he inferred from that sentence. This process was done with the objective of finding the amount of labels that ensures that the system will not be too generic, but does not complicate the training process.

The semantic value label list was compiled based on the existing library of gestures that the robotic platform can use. The gestures were analysed in order to identify the communicative goal with which they could be associated. For example, a particular gesture might be suited to accompany a utterance that seeks to greet the user, while a second expression could combine better with a sentence used to retrieve personal information from the other peer. Similar gestures were clustered together, in order to reduce the amount of possible semantic value labels. Finally, a single label was defined for each cluster.

Because the required amount of instances necessary to train the proposed models is significantly high (2600 instances each), an automated process for creating new dataset instances was developed using a python script. When this program is run, it starts by loading all the utterances in the corpus. Then, the following steps are performed sequentially:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<example id="0120">
<sentence>How was your trip? Profitable?</sentence>
<tokens>
<token id="1" lemma="how" pos="ADV" surface="How" />
<token id="2" lemma="be" pos="VERB" surface="was" />
<token id="3" lemma="-PRON-" pos="PRON" surface="your" />
<token id="4" lemma="trip" pos="NOUN" surface="trip" />
<token id="5" lemma="?" pos="PUNCT" surface="?" />
<token id="6" lemma="profitable" pos="ADJ" surface="Profitable" />
<token id="7" lemma="?" pos="PUNCT" surface="?" />
</tokens>
<intentions>
<token id="1" value="B-REQUEST_PERSONAL_INFO" />
<token id="2" value="I-REQUEST_PERSONAL_INFO" />
<token id="3" value="I-REQUEST_PERSONAL_INFO" />
<token id="4" value="I-REQUEST_PERSONAL_INFO" />
<token id="5" value="I-REQUEST_PERSONAL_INFO" />
<token id="6" value="I-REQUEST_PERSONAL_INFO" />
<token id="7" value="I-REQUEST_PERSONAL_INFO" />
</intentions>
<gestures>
<token id="1" value="B-OTHER_PEER" />
<token id="2" value="I-OTHER_PEER" />
<token id="3" value="I-OTHER_PEER" />
<token id="4" value="I-OTHER_PEER" />
<token id="5" value="I-OTHER_PEER" />
<token id="6" value="I-OTHER_PEER" />
<token id="7" value="I-OTHER_PEER" />
</gestures>
</example>
```

Figure 5.9: Example of an instance taken from the dataset developed for the gesture prediction model.

1. The utterance is passed through a Part-of-Speech tagger provided by spaCy [1], a python library for natural language processing applications.

2. The annotator is presented with the list of tokens generated, and offered the choice of correcting any mistake.

3. The list of tokens is passed through the first model, in order to obtain a list of communicative intentions.

4. The annotator is presented with the list of intentions predicted, and offered the choice of correcting any mistake.

5. The list of tokens is combined with the tokens, and then passed through the second model, which predicts the semantic values.

6. The annotator is once again presented with the list of semantic values, and offered the choice of correcting any mistake.

[1] https://spacy.io/

7. The utterance, tokens, intention labels, and semantic value labels are stored in a file.

The first instances of the dataset were handcrafted, and then used to train the models. Once a first version of the models was obtained, the program designed above was used to add new instances to the dataset. The models were retrained every few hundred instances, in order to improve their accuracy.

5.3.2.4 Result of the training process

The final version of the datasets contains 2600 instances. For training each model, its dataset has been split in three partitions: training (60%), validation (20%), and testing (20%). The validation set is used during the training process, after each epoch, to tune the model's parameters. The test dataset is used to evaluate the model once the training has been completed. The split of the dataset was performed ensuring that the representation of the possible output labels was proportionally distributed between all three subsets. This means that, for example, 60% of the instances containing a particular semantic value label will be located in the training dataset, while the remaining 40% will be split between the validation and testing datasets. The configuration for both models has been shown in Section 5.3.2.2.

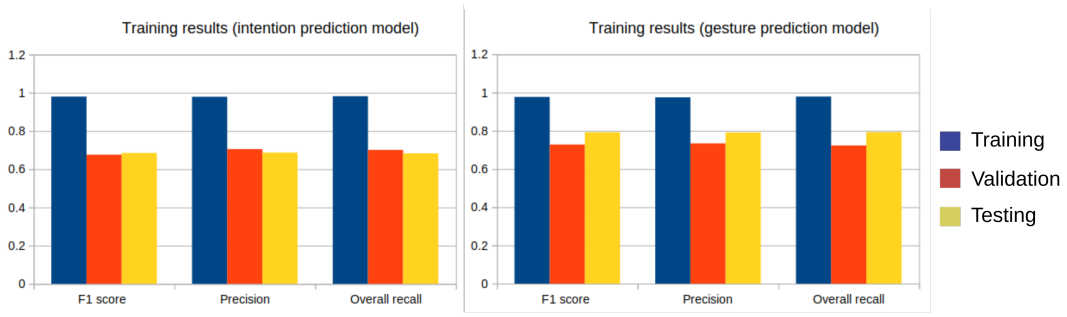


Figure 5.10: Results for the training of the models.

Regarding the intention prediction model, the training was conducted for 76 epochs. Three metrics were used to evaluate the model: (i) *precision*, which represents the number of correct predictions divided by the total amount of predictions for each label; (ii) *recall*, which represents the ratio between the correct predictions for one class and the total amount of labels from that class; and (iii) f1 score (or F-score), which represents the harmonic mean of the model's precision and recall. The best f1 score for this training was 0.98, with a validation f1 score of 0.676. Then, the trained model was evaluated using the remaining instances of the dataset, achieving a f1 score of 0.685, a precision of 0.687, and an overall recall of 0.683. The gesture prediction model was trained for 44

epochs, obtaining a training f1 score of 0.98, and a validation f1 score of 0.728. Again, the model was evaluated with the testing dataset, achieving an f1 score of 0.792, an overall precision of 0.791, and an overall recall of 0.793. Figure 5.10 shows a summary of these results.

Overall, the results of the training show that, although the models have room for improvement, their ability to predict the gestures that are better suited for accompanying a particular utterance is satisfactory, given the complexity that this task has. Also, it is important to mention that, although the model is able to predict the correct gestures 79% of the time, this does not mean that the remaining 21% of predictions lead to unnatural interactions, as usually wrong predictions tend to be gestures that also suit the utterance. For example, if the robot has to ask a personal question to the user, the correct semantic value that the model should predict is a gesture that points to the other peer, as a way to invite him/her to share that information. When the model provides a wrong prediction, it tends to select a gesture where the robot changes its posture to convey that it is asking a question. Although this gesture is not the optimal to use in this situation, it is also appropriate, and thus does not hinder the interaction.

5.3.3 Operation of the co-speech gesture prediction process

This section describes the overall operation of the proposed co-speech gesture prediction module. The module is composed by the following elements: (i) the intention and gesture prediction models described in Section 5.3.2.2; (ii) filters used to correct any possible prediction errors in the output of the models; and (iii) a gesture selection and synchronization module. The prediction models have been trained beforehand in an external machine. Because the robot is not equipped with a graphics card, both models are run in CPU.

As stated before, all communicative actions that only include a utterance are sent to the co-speech prediction module. It is important to mention that this utterance can contain multiple sentences. In that case, they would be processed together. This is because the gestures predicted for a specific sentence might be different if the sentence is preceded by other sentences. For example, if the user is asked either *"Do you like movies?"* or *"What is the last movie you watched?"*, in both cases the robot will probably perform the same expression, as they are both questions where the robot tries to obtain personal information about the user. But if the utterance combine both questions and asks *"Do you like movies? What is the last movie you watched?"*, then the robot might perform a different expression for the second part of the utterance, so it does not repeat the same gesture twice in a row. It is also possible that two consecutive sentences have independent intentions. For example, if the

robot utters *"Do you like movies? I enjoy them a lot."*, the first sentence has the intention of retrieving information about the user, while the second one has the intention of providing information about the robot. In this case, due to the different intentions of the sentences, different gestures will be used for each sentence.

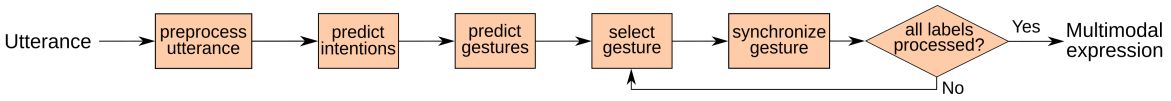


Figure 5.11: Activity diagram representing the steps followed for obtaining a multimodal expression from a utterance.

Figure 5.11 shows the complete process followed to obtain multimodal expressions based on the robot's speech. This process can be decomposed in the following tasks: (i) preprocess the utterance; (ii) predict the communicative intentions; (iii) predict the semantic values of the gestures that have to be performed alongside the utterance; (iv) select the appropriate gestures given the semantic values predicted; and (v) synchronize the gestures with the speech. The last two tasks will be performed for each semantic value predicted by the model. These tasks will be described in detail in the following sections.

5.3.3.1 Preprocessing of the utterance

The first step in the pipeline is to process the utterance that has to be sent to the TTS in order to remove special characters (except delimiting characters like . or ?). The delimiters are not removed because they can convey information that might be relevant for predicting the gestures (for example, the presence of a question mark indicates the end of a question, and thus a possible change in the intention of the robot beyond that point). Next, contractions have to be processed in order to separate negation adverbs or contracted verbs. For example, *won't* has to be transformed into *will n't*, *where's* into *where 's*, and *hasn't* has to be transformed into *has n't*. Once the text is ready, it is passed through the spaCy Part-of-Speech tagger. This process is shown in Figure 5.12. The sequence of PoS tags is Verb (VERB), adverb (ADV), pronoun (PRON), verb (VERB), noun (NOUN), and punctuation sign (PUNCT).

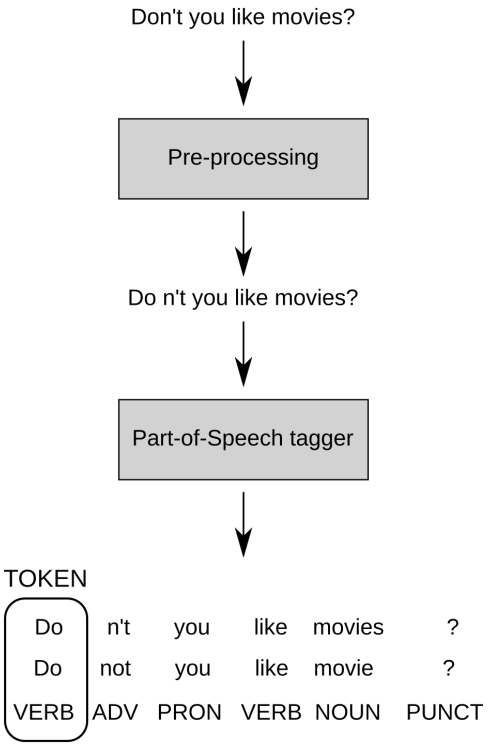


Figure 5.12: Diagram representing the pre-processing stage.

5.3.3.2 Prediction of the robot's communicative intention

The result of the first stage of the process is an array that contains individual tokens for each word in the robot's utterance (contractions that were divided into two different text chunks are considered separate tokens). Each token contains the word and its lemma, as well as the Part-of-Speech label assigned. For example, for the utterance *"Don't you like movies?"*, the sentence would be split in the following words: *Do n't you like movies ?*, the lemmas would be *Do not you like movie ?*, and the Part-of-Speech labels would be *VERB ADV PRON VERB NOUN PUNCT*. This array is sent to the first model in order to predict the intention or intentions that the robot has. The process followed to obtain this prediction has been described in Section 5.3.2.2. The list of possible labels, as well as a description of their meaning is shown in Table 5.3.

Label	Description
State Robot Fact	The robot presents information relative to itself.
Request Personal Info	A question oriented to retrieve information about the user.

Table 5.3: Labels depicting the possible communicative intentions that the robot might use.

Label	Description
Explain	Generic utterance that serves for exposition purposes.
Ask Question	Generic question that is not about any personal information.
State Third Party Fact	Statement that presents facts about a third party not involved in the interaction.
State User Fact	Statement that presents a fact about the user.
Voice Opinion	The robot expresses an opinion that is not about the user.
Request Action	The robot asks the user to do something (it can be anything from a polite request to a command).
Agree	The robot is agreeing with the previous statement uttered by the user.
Ask For Opinion	The robot asks the user to provide his/her opinion about something.
Refuse	The robot provides a negative answer to a question from the user, or rejects the previous statement.
Emphasize	The robot provides information about which it feels strong (it does not consider sentiment, it can be both positive or negative).
Notify Action	The robot lets the user know about a present or future action that it will carry on.
Counter Opinion	Accompanying an *Explain* Intention, it aims at providing a counter opinion to the one presented in the first part of the utterance.
Apologize	The utterance is used to show remorse (it does not need to be an explicit apology).
Express Doubt	The robot provides a fact about which it is not completely sure.
Encourage	The robot aims at encouraging the user to do something.
Greet	Utterance used to initiate or end an interaction.
Confront	The robot is engaged in a heated argument with the user.
Show Empathy	Utterance used to acknowledge and show empathy towards the user's last statement.

Table 5.3: Labels depicting the possible communicative intentions that the robot might use.

Label	Description
Thank User	Utterance used to show gratefulness towards the user.
Calm User	The robot tries to calm the user or state a fact that calms the user.
Make Offer	The robot offers to do something for the user.
Express Need	Utterance used to express a need so the user can act on it.
Express Enthusiasm	The robot states a fact about which it feels strong. Opposite to the *Emphasize*, it has a positive sentiment.
Offer Choice	Utterance used to offer the user a choice among multiple options.
Give Evaluation	The robot conveys an opinion about the user.

Table 5.3: Labels depicting the possible communicative intentions that the robot might use.

In the example presented above, the intention of the *"Don't you like movies?"* is to obtain personal information about the other peer in the interaction, so the sequence of communicative labels predicted should be *B-Request Personal Info, I-Request Personal Info, I-Request Personal Info, I-Request Personal Info, I-Request Personal Info, I-Request Personal Info, I-Request Personal Info*.

In an ideal situation, the distribution of the intention labels would be uniform, with clearly defined boundaries between them. But due to potential errors in the prediction process, there can be situations where the label assigned to a token is different from both the previous and the next labels in the sequence. For example, while the correct label sequence for the *"Don't you like movies?"* is *B-Request Personal Info, I-Request Personal Info, I-Request Personal Info, I-Request Personal Info, I-Request Personal Info, I-Request Personal Info*, prediction errors could cause this sequence to be something like *B-Request Personal Info, I-Request Personal Info, I-Request Personal Info, B-Ask For Opinion, B-Request Personal Info, I-Request Personal Info*. This example is depicted in Figure 5.13. Errors in the prediction of intentions could propagate to the prediction of gesture semantic values, and those could cause problems during the gesture selection and synchronization phase. In order to reduce the effect that they have on the overall result of the process, a filter was added to the prediction step. This filter works on two assumptions: (i) tokens with the same labels tend to be located next to each other, (ii) there is no situation in which one label will appear only once in an utterance. The filter performs the following steps:

1. For each different label, the algorithm stores all the positions in the predicted labels set in which they appear.
2. If all the positions in which a label appears are consecutive, then that sequence of positions is considered a *closed cluster*.
3. If they are non consecutive, the algorithm tries to split that sequence of positions for that label into subsequences of consecutive positions. Any subsequence larger than a threshold is considered a *closed cluster*. Otherwise, it is classified as an *open cluster*
4. Every label in a position included in a *closed cluster* is assumed to be correctly predicted, and these clusters are stored in a list L
5. The *open clusters* are sorted based on their length
6. The largest *open cluster* A is extracted. Its lowest and highest positions will be named A_l and A_h, respectively.
7. If none of the correctly predicted labels fall in between A_l and A_s, that *cluster* is considered closed, and the positions in A are added to L
8. The algorithm repeats the previous 2 steps for all *open clusters*.
9. All the clusters in L are sorted based on their lowest position (the positions inside each cluster are already ordered).
10. For every position i in L that do not have a correct prediction, the algorithm assigns it the label from position $i - 1$.

Figure 5.13 depicts the process followed for predicting the communicative intention/s of a utterance and filter the prediction obtained to correct any possible error.

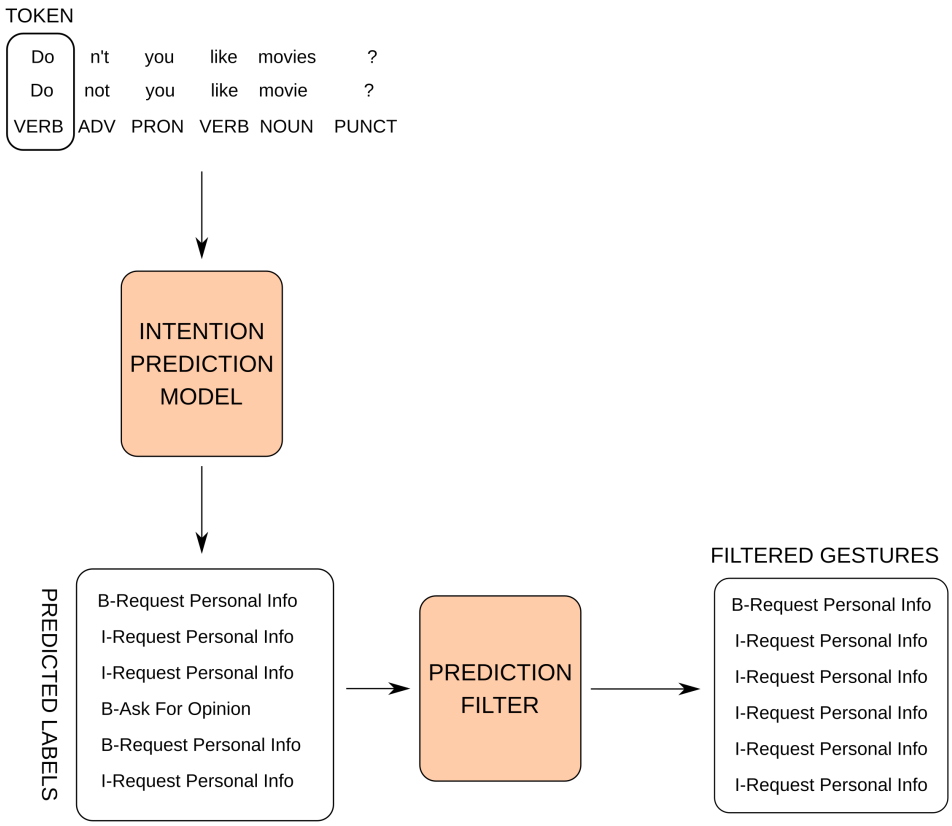


Figure 5.13: Diagram representing the prediction of intentions.

5.3.3.3 Gesture prediction

The filter presented in the previous section solves the problem of having individual labels scattered throughout the predicted sequence. The output of the filter is then packaged with the tokens containing the words and the Part-of-Speech labels and sent to the model for obtaining the semantic values for the gestures that have to be added to the speech. The prediction process is described in Section 5.3.2.2. The possible semantic values of the gestures that can be paired with an utterance are shown in Table 5.4.

Label	Description
Other Peer	Extending the arm towards the other peer with the back of the hand facing him/her.
Explain	Gesture designed for generic situations.

Table 5.4: Labels depicting the possible semantic values that can be conveyed with the robot's gestures.

Label	Description
Self	Reflexive gesture where the speakers points towards himself/herself with the palm of the hand.
No	Head shake to convey negation.
Question	Gesture or posture adopted for asking a generic question.
No Gesture	No action should be performed.
Yes	Head nod to express an agreement.
Neutral	Gesture that expresses doubt, usually by shrugging the shoulders.
Front	Pointing gesture with an extended finger towards the other peer.
Emphatic	Variation of the explain gesture with a higher intensity.
Please	Posture or gesture adopted for requesting help to the other peer.
Sorry	Posture or gesture to show remorse.
Calm Down	Gesture used to induce calm in the other peer.
But	Change in stance that punctuates an idea opposing to what has been conveyed in the first part of the sentence.
Third Person	Arm extended in a perpendicular direction to the other peer, in order to make a reference about an external party.
Come On	Gesture oriented to incite the user to do something, either by himself/herself or with the robot.
Greet	A gesture used to start or finish a conversation.
Thanks	A gesture used to show gratefulness.
Iterate	Rhythmic beats used to punctuate the items in an enumeration.
Enthusiastic	Display of strong positive affect towards the content of the utterance.
Thinking	Gesture used to represent the process of thinking about the information being conveyed.

Table 5.4: Labels depicting the possible semantic values that can be conveyed with the robot's gestures.

Continuing with the previous example, the correct sequence of semantic values for the utterance *"Don't you like movies?"* should be *B-Other Peer, I-Other Peer, I-Other Peer, I-Other Peer, I-Other Peer, I-Other Peer*. The output generated by the second model is also passed through the filter described above to correct any possible prediction errors, as shown in Figure 5.14. The corrected sequence is then sent to the next method of the pipeline: the gesture selection method.

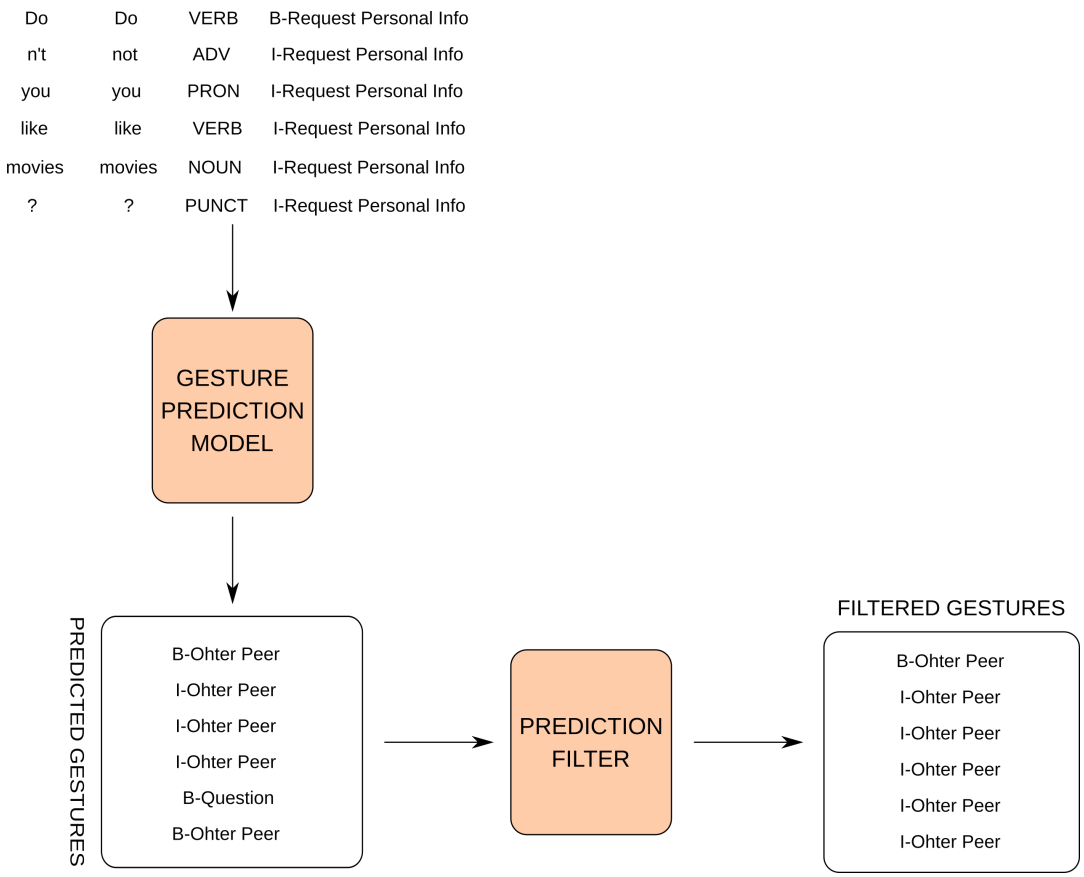


Figure 5.14: Activity diagram representing the gesture prediction stage.

5.3.3.4 Selection of gestures based on their semantic values

Once the sequence of semantic values has been obtained, the co-speech gesture prediction module has to select the appropriate gesture/s from the expression library. This process is described in Figure 5.16 First, the utterance is again processed to remove all special characters. In this case, this includes also all punctuation signs. Then, the duration of the sentence is estimated. Sentences of different durations were sent to the TTS module, and their duration was measured. Based on these durations, a factor was computed to model the relation between the number of characters in an utterance and

its duration. From here on, this factor will be referenced as the character/time factor. Then, the utterance is split into chunks according to the labels predicted. Each chunk would then correspond to a segment of the utterance that has been tagged with the same label (if the same label is assigned to two separate parts of the utterance with other labels in between, then they would be split into two different chunks). The starting point for each of the generated chunks would be computed as the amount of characters that precede it. Continuing with the example set in the previous section, the utterance *"Don't you like movies?"* has been labelled with a single label: Other Peer, and thus the utterance would be processed entirely. If instead, the sentence that the robot has to utter is *"What is your name? My name is Mini"*, the communicative intentions would be to retrieve personal information about the user with the question, and then provide personal information about the robot with the statement. The semantic values predicted by the model would be *Other Peer* and *Self*. This would lead to the creation of two chunks: (i) the first one corresponds to the *What is your name?* question, which requires a gesture with the semantic value *Other Peer*; and (ii) the second one corresponds to the *My name is Mini"* statement, which requires a gesture with the semantic value *Self*.

Once the utterance has been split, the co-speech gesture prediction module selects and synchronizes gestures for each chunk individually. This is performed according to a series of handcrafted rules stored in an external file. This approach allows developers to customize the synchronization process based on the particularities of their robotic platforms, and to use different criteria for each possible semantic value labels. Each rule contains the following fields:

- **synchronization type:** Indicates which feature should be used for synchronizing speech and gesture. If *position* is selected, then the gesture is tied to a specific point in the utterance chunk. If either *PoS* or *word* are selected, then the start of the gesture will be connected to the appearance of a specific Part-of-Speech label or word.

- **synchronization value:** Defines the synchronization point, based on the type defined in the previous field. For positions, it allows to tie the start of the gesture to the beginning of the chunk, the middle, or connecting the ends of both the chunk and the gesture. For Part-of-Speech or words, the actual label or word can be defined.

- **synchronization point:** Allows to define a finer synchronization point. For example, if the start to the gesture has to be tied to a specific Part-of-Speech but it appears more than once in the chunk, it allows to define to which appearance should be connected.

- **offset:** Allows to define an offset in characters (either positive or negative) from the point described by the previous fields.

- **mode:** Defines the execution mode of the gesture. Under *standard* mode, the gesture is performed only once. Under *symmetric* mode, the algorithm concatenates symmetric versions of a gesture. Finally, under *continuous* mode, the algorithm fills the utterance chunk with as many repetitions of the gesture as possible.

The synchronization modes have been developed to give more freedom to the developers on how to adapt the selection and synchronization processes to the particularities of a given type of gestures. For example, if a semantic value is connected to a static pose, or an emblematic gesture that conveys a specific message according to a series of conventions, then it might be OK to perform the gesture only once. On the other hand, if a semantic value is connected to a gesture where the robot performs a repetitive motion, then it might be better that the motion extends during the whole speech chunk. The *standard* mode was designed for the first situation described, while the *continuous* mode covers the second situation. Finally, a third synchronization mode was designed for situations where the robot has to perform an expression that can be decomposed in a sequence of symmetric motions. An example of this would be a utterance that provides a list of items, and an expression that punctuates each item in the list with an arm motion, alternating between both arms.

Developers can define multiple rules for each possible label, and the algorithm will check them in the order in which they are stored in the rule file. If the rule being checked is met, then the selection is performed. For example, a particular rule might require that the beginning of the gesture is connected to the first appearance of an adverb. Thus, the rule can only be met if the speech chunk contains at least one adverb. If this rule cannot be met, the algorithm checks the next rule. If no rule can be followed, then the label is discarded, and no gesture is attached to that specific utterance chunk. Usually, a reason for this failure is to have a utterance chunk that is too short, which makes impossible to fit any gesture into it. For example, the *No* gesture could have one rule specifying that it should start when the actual word "no" is uttered, while a second rule could specify that the gesture should start at the beginning of the chunk. The selection method would then first look for the adverb "no" in any chunk that has been tagged with the *No* label, and then would try to add the gesture at the beginning of the chunk, if the word is not present. If the chunk is too short for the gestures attached to that label, then no gesture would be selected. The rules that would be developed for this example can be seen in Figure 5.15.

```
<?xml version="1.0" encoding="UTF-8"?>
<rules>
<NO>
    <rule sync_type="word" sync_value="no" sync_point="first" offset="0" mode="standard"/>
    <rule sync_type="position" sync_value="start" sync_point="first" offset="0" mode="standard"/>
</NO>
</rules>
```

Figure 5.15: Example of synchronization rules for a semantic value.

The selection method receives the label, the list of words in the chunk, their Part-of-Speech value, and the starting point (in number of characters). A limited set of expressions have been developed for each possible semantic value. Gestures were created through a crowd-sourcing process. In this process, participants were seated in front of Mini. Then, the interviewer presented a definition of each semantic value and asked the participants to explain how Mini could express that particular idea. All the participants had previous experience working with Mini, and thus were familiar with its expressiveness capabilities. Then, for each semantic value, the expressions described by the participants were compared, in order to identify the common features. Finally, these features were used to design the library of expressions used by the co-speech gesture prediction module.

During the selection process, the gestures corresponding to the semantic value label attached to the speech chunk are retrieved, along with their duration. Based on the set of rules for that particular semantic value label, the method computes the starting point of the gesture in the chunk, and finds the time window in which the gesture has to be performed. This window is computed as the difference between the final point of the chunk and the starting point of the gesture, and then transformed into time by multiplying the amount of characters by the character/time factor. If the execution mode defined by the rule is *symmetric*, the time window should be big enough to fit at least one instance of each symmetric gesture. For example, if the robot has to utter a list of elements, the algorithm could tag that utterance whith the *Iterate* label. The gesture attached to this label has the robot punctuating an item of the list with an arm beat. If the rule associated to the *Iterate* label uses the *symmetric* mode, this means that the utterance should be accompanied by gestures that perform arm beats with alternate arms. Thus, when measuring if the gesture can be performed in the available time window, the algorithm should check that the robot is able to perform at least a beat with each arm. The gestures that fit in that time window are stored in a set. Finally, the selection method draws randomly one of the gestures stored in the set, and passes it to the synchronization method. Figure 5.16 shows the steps performed during the gesture selection process.

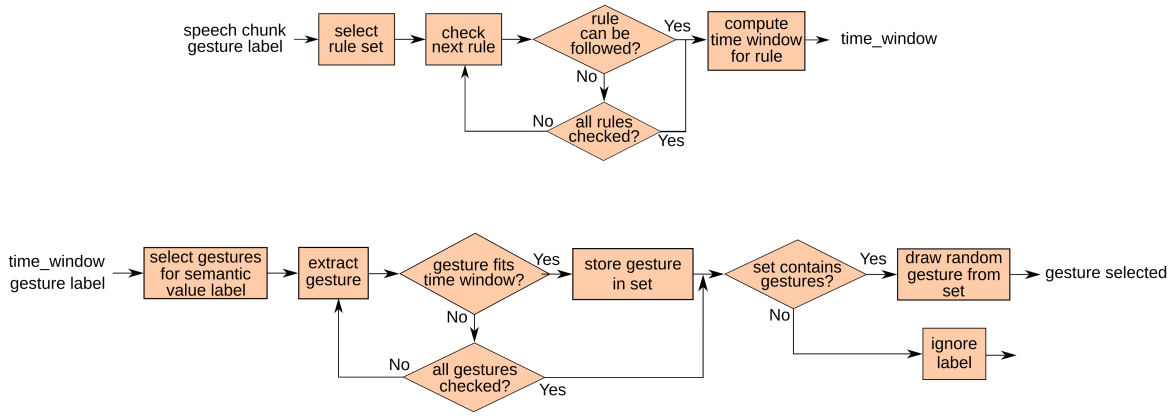


Figure 5.16: Activity diagrams representing the selection phase. The diagram at the top represents the process followed for computing the time window in which gestures can be performed, while the diagram at the middle represents the process for selecting the gesture based on the time window computed.

5.3.3.5 Synchronization of speech and gestures

The last part of the process is synchronizing the gesture selected with the speech chunk. This process is shown in Figure 5.17. The synchronization algorithm has to compute the amount of gestures that have to be performed in the time window found during the selection phase. If only one gesture has to be performed, the start point is computed based on the rule selected. On the other hand, if more than one gesture have to be synchronized with the utterance chunk (if the synchronization mode defined by the rule is either *symmetric* or *continuous*), the algorithm computes the amount of gestures that can be performed in the available time window, and the start point for each one, in seconds. The result of the synchronization process is a list of the gestures that have to be performed, and the starting point in seconds for each of them.

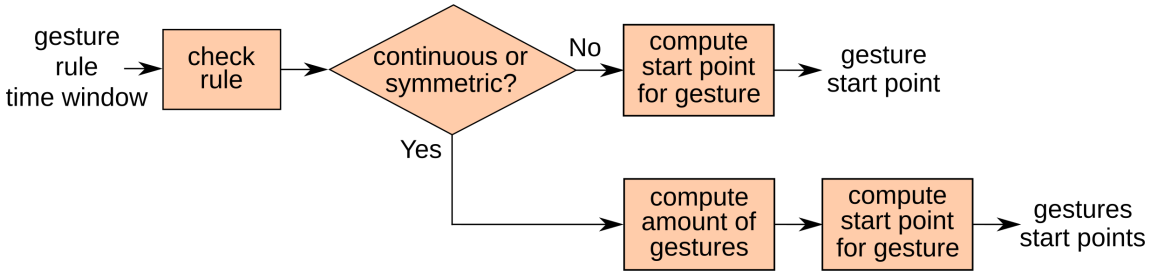


Figure 5.17: Activity diagram representing the synchronization of a gesture and a speech chunk.

The last step performed by the co-speech gesture prediction module is to create the multimodal gesture that will be returned to the Expression Executor to be executed. This gesture contains the utterance and the gestures predicted by the co-speech gesture prediction module. Each gesture includes a delay that allows to configure its starting point. For the sake of synchronization, the delay for each gesture starts counting when the TTS receives the utterance. The Expression Executor then configures a gesture with the utterance and the sequence of gestures, and executes it following the procedure described in Section 4.5.3.

5.4 Evaluation of the proposed system

This section presents the evaluation conducted in order to test both the pulse-based liveliness module and the co-speech gesture prediction module presented in Section 5.3. First, as in previous chapters of this thesis, a series of objective metrics will detail the performance of the proposed approaches. Next, a case of study is presented in order to show how the proposed modules work when the robot is performing one of its tasks, and under which conditions each liveliness approach is used.

5.4.1 Objective evaluation

The first part of the evaluation will consist on an objective analysis of the proposed implementation of both liveliness approaches. The goal is to evaluate if the module can be embedded in a robotic platform under real conditions. This analysis will include: (i) the hardware resources used, namely the CPU and RAM that both modules require, in order to understand if they can be run alongside the rest of the architecture; and (ii) the time required by the co-speech gesture prediction module to generate a multimodal expression from a utterance, decoupled into the different subprocesses involved in the prediction and synchronization pipeline. The pulse-based liveliness module is only included in the resource usage analysis, but not in the other two. This is due to the fact that the behaviours generated by this module do not depend on timing for achieving their task, and instead can be performed at any given time. Thus, they are not constrained by the temporal dimension of human-robot interactions.

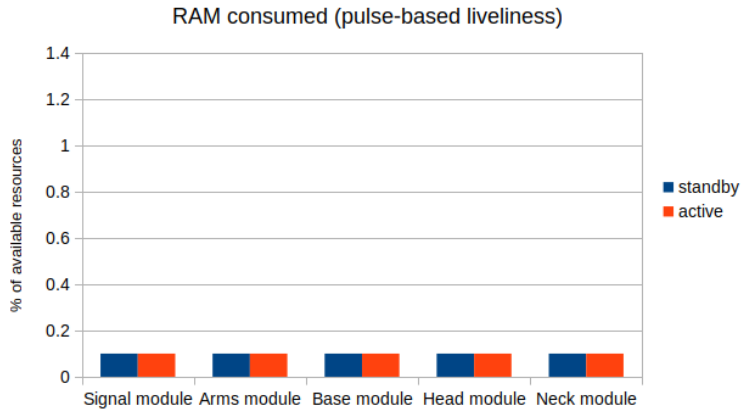


Figure 5.18: Peak use of RAM by the pulse-based liveliness module.

5.4.1.1 Resource usage

This section presents the evaluation of the performance of the proposed liveliness approaches. As stated in Section 5.3, the pulse-based liveliness module is divided in multiple sub-modules. One of them manages the generation of the signal that will represent the robot's pulse, while individual modules receive the samples extracted from the signal and use them to generate unimodal expressions. This leads to a total of 5 modules (signal, arms, head, neck, base). Regarding the co-speech gesture prediction module, it has been developed as a single module. For both modules, the temporal analysis has been conducted under two different conditions: (i) under the *standby* condition, the robot stays in a standby state, without performing any expressions; and (ii) under the *active* condition, the modules are in operation (generating liveliness behaviours or creating multimodal expressions from a speech input). In both cases, the software architecture presented in Chapter 2 is running in the robot. As a reminder, Mini has 16 GB of RAM, and an Intel i5-3550 CPU, with four cores running at 3.3 GHz. Mini's operating system is Ubuntu 16.04 64 bits.

Regarding the use of memory, it was observed that it is kept constant under both conditions for all the modules involved, while small variations on CPU use where observed for the pulse-based liveliness module. Each sub-module is consuming a 0.1% of the available RAM, as shown in Figure 5.18 The signal module consumes between 0.7 and 1.3% of the robot's processing capacity, while the CPU used by the remaining modules is in between 0.0% and 0.7%. The CPU use by the pulse-based liveliness module can be seen in Figure 5.19. On the other hand, the co-speech gesture prediction module uses around the 8% of the available RAM, and consumes most of the processing capacity of the robot (351% of CPU, which translates in the complete use of three CPU cores and half of the other). These results, which can be observed in Figure 5.20, show that the pulse-based liveliness

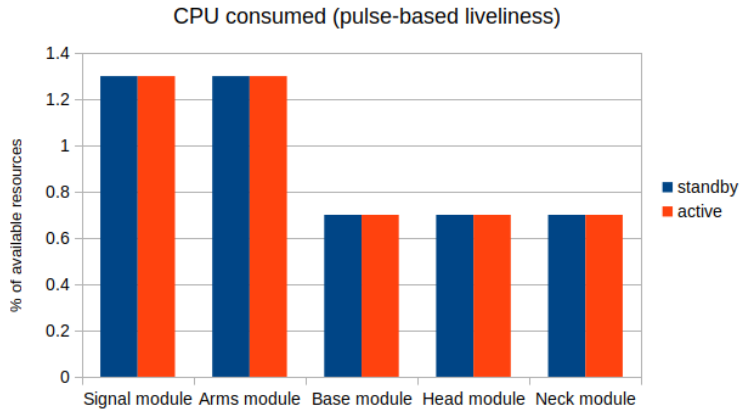


Figure 5.19: Peak use of CPU by the pulse-based liveliness module. The results are shown as a percentage of the processing capacity of a single CPU core.

module can be perfectly integrated in the robot's architecture, and run alongside all the other modules. However, the required amount of CPU is a significant constraint for the integration of the co-speech gesture prediction module in the regular architecture of the robot, at least with the current hardware. This was not a surprising result, as the co-speech gesture prediction module was at a disadvantage. Deep learning models tend to have a high computational load, and in many cases are run directly on GPUs, instead of using the CPU. Mini's hardware is not designed with this type of models in mind. Thus, it was expected that the co-speech gesture prediction model would have high demand of processing capacity, A possible solution for this problem is to run this model in an external hardware.

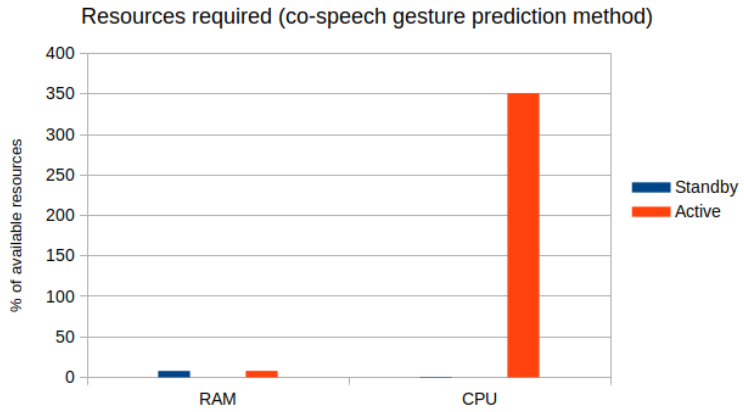


Figure 5.20: Peak use of resources by the co-speech gesture prediction module under two conditions (with the module in standby, and with the module working). The results for the CPU usage are shown as a percentage of the processing capacity of a single CPU core.

5.4.1.2 Response time for the co-speech gesture prediction module

This section presents the response times extracted for the different subtasks that are performed by the co-speech gesture generation module. Because this approach has been integrated in the process for conveying communicative messages to the user, the response time for the prediction module has to be added to the delays generated by the HRI Manager and the Expression Manager, which have been discussed in Sections 3.6.2.2 and 4.6.1.2, and compared to the threshold that represents the limit for an acceptable pause in an interaction. Previous chapters of this manuscript have established that, while some researchers follow the rule known as the *"Two second" rule* and consider that two seconds is the amount of time after which messages might loose their meaning, a more strict threshold of 1 second will be enforced, to ensure a more agile interaction. In the objective evaluation of the proposed co-speech gesture generation module, the response time will be compared to both thresholds, using the *"Two second" rule* as the maximum delay that can be accepted. This will be the only threshold considered due to the fact that the proposed approach will only be used to enhance actions intended to convey messages to the user.

The proposed module for predicting gestures is prepared to receive utterances of different lengths, even multiple sentences at once. In order to conduct a proper temporal analysis, the module has been tested under three different conditions: (i) a single short sentence is passed (8-word sentence); (ii) a medium-sized utterance composed of several sentences is passed (30-word utterance); and (iii) a long-sized utterance with multiple sentences is passed (60-word utterance). Under all three conditions, the prediction process has been divided in the following subtasks, which are performed sequentially:

- **Pre-processing:** The preparation that is required before sending the utterance to the intention prediction model. It involves the process described in Section 5.3.3.1.
- **Predict intentions:** The process for extracting the communicative intention from the utterance. It involves the prediction process described in Section 5.3.3.2.
- **Filter intentions:** The filtering that is performed for correcting the sequence of labels generated by the intention prediction model.
- **Predict gestures:** The process for obtaining the semantic values of the gestures that have to be combined with the utterance. It involves the prediction process described in Section 5.3.3.3.
- **Filter gestures:** The filtering that is performed for correcting the sequence of labels generated by the gesture prediction model.

- **Gesture preparation:** Preparation stage for the synchronization of speech and gestures. The sequence of labels is converted into a key-value pair array, were the keys are the gestures that have to be added to the utterance, and the values are the synchronization points (multiple instances of one gesture can appear in different parts of an utterance).
- **Gesture synchronization:** Process that loads the appropriate gestures from the expression library and computes their start point (time that has to pass from the beginning of the utterance, in seconds).
- **Sending expression:** Combines the utterance and the gestures that have been selected into a single expression request message, that will be processed by the Expression Manager as any regular request coming from the rest of the architecture.

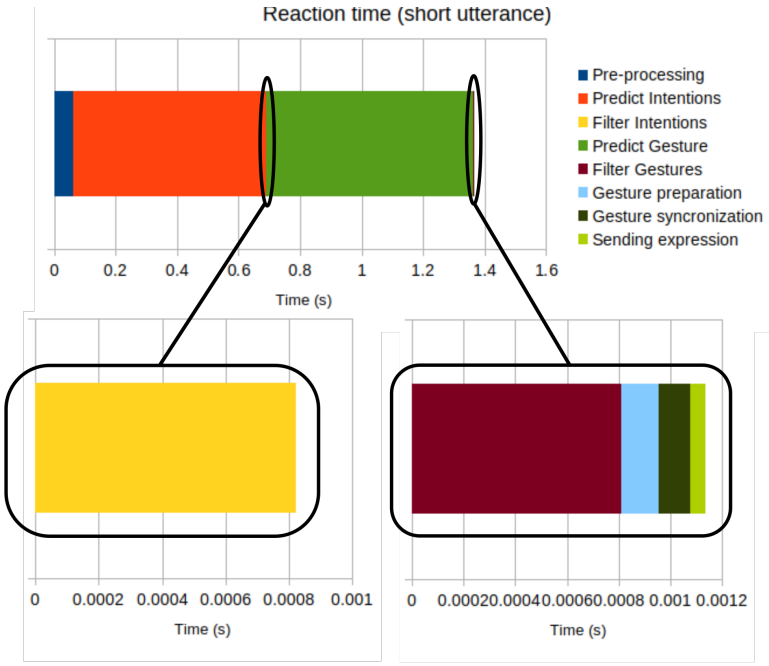


Figure 5.21: Duration of the different sub-processes involved in the generation of multimodal expressions with synchronized verbal and non-verbal behaviour, when a short sentence is passed as input. Top chart represents the complete process, while the low chart shows those process that are too short to be appreciated in the top chart. Bars represent the average duration for each process.

For each trial, the entire process will be performed 10 times for the same utterance, and the average duration for each of the subtasks will be obtained. The Figures 5.21, 5.22, and 5.23 show the results extracted from the temporal analysis for a short, medium, and long utterances, respectively. Because completing some of the described subtasks takes a time that is several orders of magnitude

smaller than others, three charts are included in each Figure. Chart on top shows the combination of all the processes, while the charts on the bottom show in more detail those processes that are too short to be appreciated correctly. Additionally, Figure 5.24 shows a comparison for the full reaction time for all three conditions. This reaction time is computed as the sum of the time required for completing the entire process.

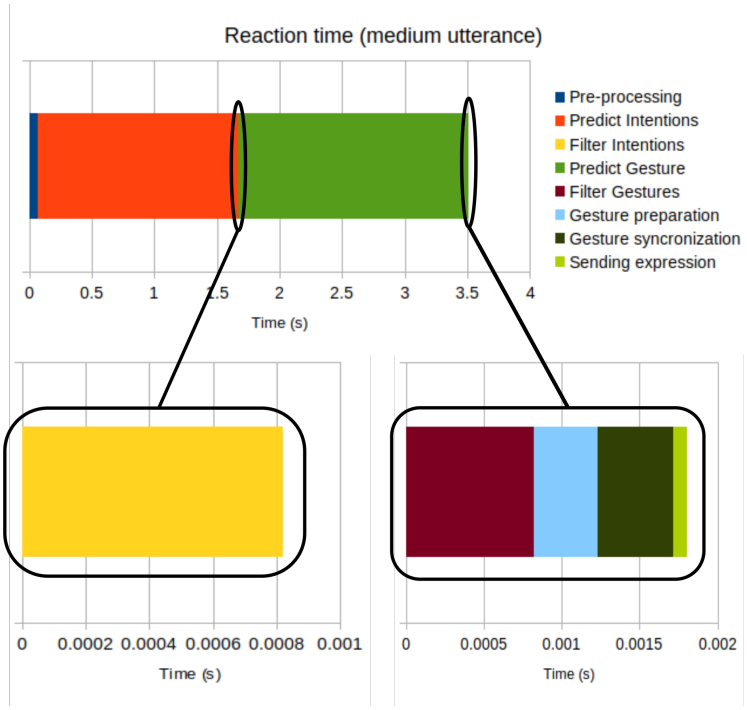


Figure 5.22: Duration of the different sub-processes involved in the generation of multimodal expressions with synchronized verbal and non-verbal behaviour, when a medium-sized sentence is passed as input. Top chart represents the complete process, while the low chart shows those process that are too short to be appreciated in the top chart. Bars represent the average duration for each process.

Figures 5.21, 5.22, and 5.23 show that both prediction steps are the clear bottlenecks of the process, and that the time required to complete them is directly proportional to the amount of tokens in the input passed to the model. This relationship is close to being linear, which allows to estimate approximately the time required to generate the prediction given the size of the input. Also, because the prediction of gestures involves the use of extra information (the sequence of intention labels), is not surprising that the time required for predicting gestures is higher than for predicting intentions (this difference increases with the size of the utterance). The other subtasks that show the highest variation are the initial preprocessing stage, the preparation step before the synchronization of speech and gestures, and the synchronization itself. Again, this is expected, as the longer the

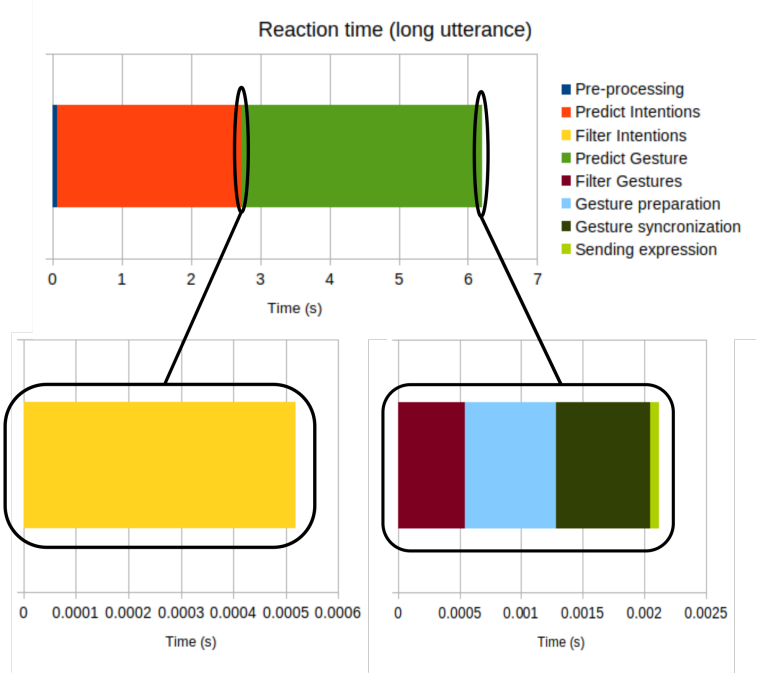


Figure 5.23: Duration of the different sub-processes involved in the generation of multimodal expressions with synchronized verbal and non-verbal behaviour, when a long sentence is passed as input. Top chart represents the complete process, while the low chart shows those process that are too short to be appreciated in the top chart. Bars represent the average duration for each process.

utterance is, the more words that have to be tagged with their Part-of-Speech labels, and the higher amount of gestures that will have to be added. However the filtering process is not affected by the number of labels to check, and, in fact, this time showed to be shorter for the long utterance. A possible explanation for this is that the filtering process is more affected by the amount of possible prediction errors present in the label sequence that by the length of the sequence itself. So, if for a longer utterance the predictions obtained are better, then the filtering process should be completed faster. Finally, the time required to build the expression request message remains mostly unaffected by the amount of gestures that have to be added to the message (there are variations, but very small).

The temporal evaluation shows that, with the current hardware installed in Mini and the co-speech prediction model running in the robot, the threshold of 1 second cannot be achieved, while staying below 2 seconds is going to depend on multiple factors. In Section 4.6.1.2, it was presented that the worst case scenario observed for delays in an interaction was around 1.16 seconds (combining the reaction times for both the HRI Manager and the Expression Manager). This time under normal conditions could be assumed to be around 1 second, as the delay for the HRI Manager was clearly the worst situation observed for a particular CA. In any case, the lowest response time

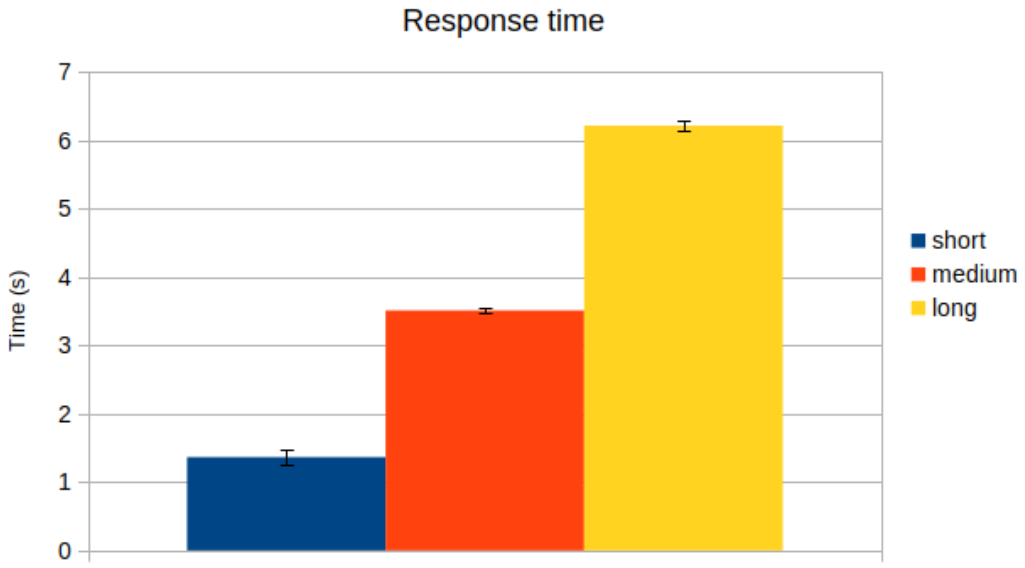


Figure 5.24: Comparison of the response time for utterances of different lengths.

observed for the co-speech prediction module was around 1.36 seconds, for the short utterance (8 tokens). This means that, added to the delays generated by the Expression Manager and the HRI Manager, the total time for performing a communicative action would be around 2.3 seconds, over the threshold defined by the *"Two second" rule*.

There are several solutions that could be tried in order to improve this response time. First, one of the biggest problems is the need for dividing the pipeline into two prediction steps: intentions and gestures. The former was included to improve the accuracy of the latter, as the intentions are more generic and easier to predict based on the input utterance. Thus, if the amount of gesture labels is reduced, the difficulty inherent to the gesture prediction process could be lowered enough to make the intention prediction unnecessary. This would have the disadvantage of making the resulting expressions more generic, and less adapted to the particularities of the input utterance. A second possible solution would be to extract the prediction model to an external server that has more powerful hardware, and thus can perform the prediction task faster. One last possibility would be to improve the hardware of the robot, although this might not be an option if the robot is a commercial platform.

5.4.2 Case of use

The integration of the proposed liveliness architecture will be described through a case of use that showcases how the co-speech gesture prediction module enhances the robot's utterances with non-verbal gestures, and how the behaviours generated by the pulse-based liveliness module are executed alongside those coming from the HRI Manager. The proposed case of use describes the same situation presented in the subjective experiment depicted in 4.6.3, where an user plays a quiz game with the robot Mini. This case of use will focus on how Mini selects and performs expressions coming from different sources. Figure 5.25 depicts the complete case of use, along with the different sources of expressions considered.

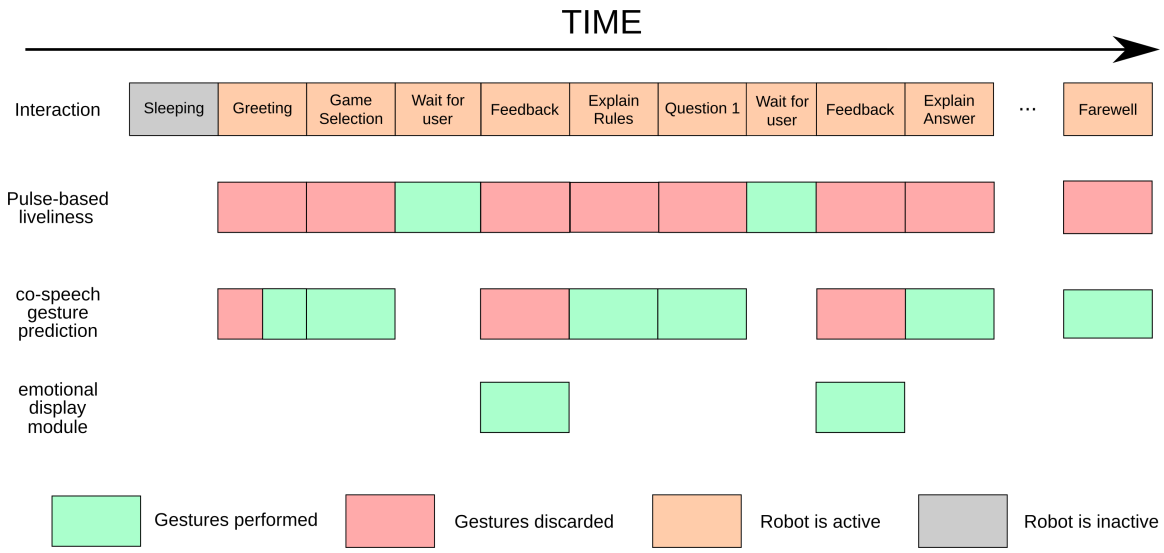


Figure 5.25: Interaction depicted in the case of use, along with the three sources of expressions considered (pulse-based liveliness, co-speech gesture prediction module, and emotional display module). The diagram indicates where the expressions that the robot performs at every time step are coming from.

Initially, Mini is placed over a table, and starts in a standby state where the robot conveys the appearance of being asleep. In this state, both the amplitude and the frequency of the signal used for generating the robot's liveliness behaviours are close to 0, so the robot does not perform any action while being asleep. When the user enters the room and strokes Mini's shoulder, the robot wakes up. This triggers a change in Mini's internal state, and thus in the control parameters for the pulse-based liveliness. In this case, because Mini is not in any particular affect state, the amplitude is set to 50%, while the frequency is set to 100%. During the change of state from sleeping to awake, Mini performs a *wake up* expression, which has the robot raise both arms and throw the head backwards, yawn

while opening the eyes, and then utter *"I was having such a good sleep"*. While this expression is being performed, the pulse-based liveliness starts generating behaviours for all the robot's communicative interfaces. But because the *wake up* gesture is using both arms, the voice, head, and eyes, all the behaviours generated by the liveliness for these interfaces will be discarded, as they have low priority.

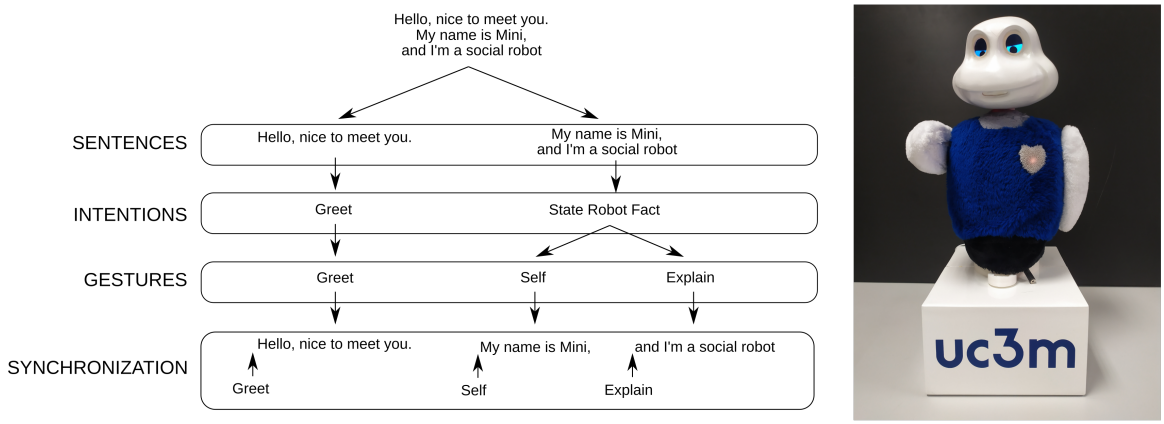


Figure 5.26: Process followed by the co-speech gesture prediction module follows for creating a multimodal expression for greeting the user. The image on the right shows Mini in the process of performing one of the selected expressions.

Next, the robot introduces itself, sending the utterance *"Hello, nice to meet you. My name is Mini, and I'm a social robot"*. Because the expression request contains only a verbal message, with no non-verbal actions attached, the Expression Manager sends this utterance to the co-speech gesture prediction module (see Figure 5.26). The module predicted two different communicative intentions: while the first sentence has the goal of *greeting* the user, the second one is used to present personal information about the robot. Thus, the output of the intention prediction model is a sequence with two labels: *Greet* and *State Robot Fact*. Based on these intentions, the gesture prediction model predicts three different semantic values: *Greet* is attached to the sentence *"Hello, nice to meet you."*, *Self* is attached to the substring *"My name is Mini,"*, and finally, *Explain* is attached to the substring *"and I'm a social robot"*. The current version of the synchronization rules state that the gestures for all possible semantic values have to be synchronized with the beginning of the corresponding speech chunk. Finally, the verbal message and the gestures selected by the co-speech gesture prediction module are performed.

After introducing itself, the robot asks the user to select what he wants to do. This is done by uttering the request *"Please, select one of the options shown in the menu"*, while showing a menu in the

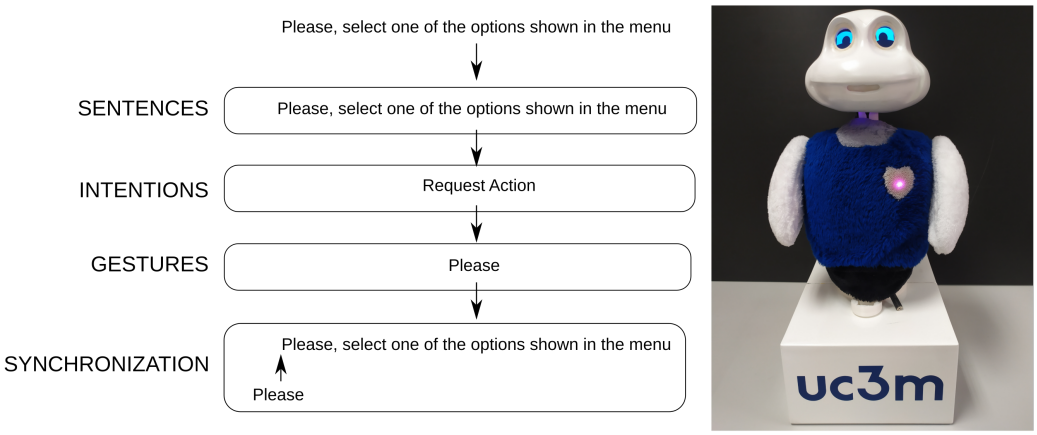


Figure 5.27: Process followed by the co-speech gesture prediction module follows for creating a multimodal expression for asking the user to select an activity. The image on the right shows Mini in the process of performing the selected expression.

tablet. Again, because the message of the robot is conveyed exclusively through verbal means (the menu displayed in the tablet is not considered part of the message conveyed), the utterance is sent to the co-speech prediction module (see Figure 5.27). In this case, the communicative intention is *Request Action*. Based on how the request is built (asking politely), the semantic value predicted for this sentence is *Please*, and once again is synchronized at the beginning of the utterance. While the user is evaluating which option to select, the pulse-based liveliness module keeps generating behaviours. In this case, because the robot is just waiting, these behaviours are going to be performed, so Mini will keep moving and changing its gaze while the user meditates his choice.

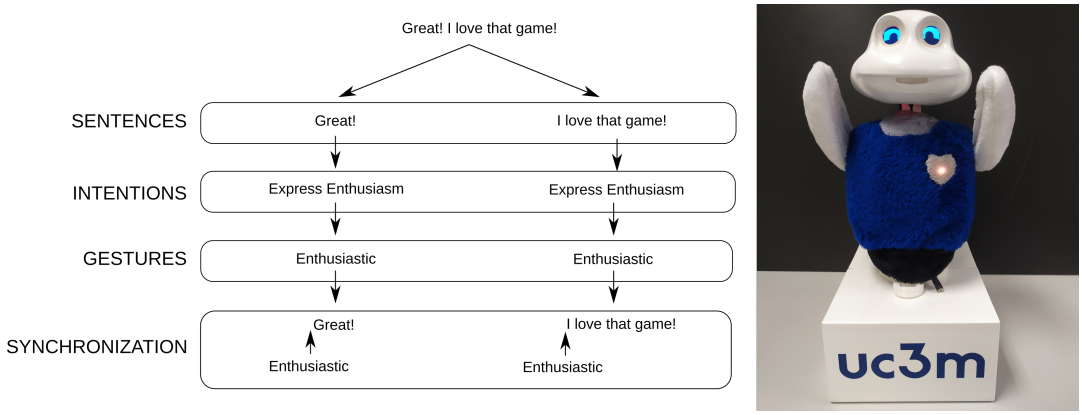


Figure 5.28: Process followed by the co-speech gesture prediction module follows for creating a multimodal expression for showing enthusiasm after the user selected the quiz game. The image on the right shows Mini in the process of performing one of the selected expressions.

Once the quiz game has been selected, the robot celebrates this decision with the utterance *"Great! I love that game!*. This utterance is passed to the co-speech gesture prediction module (see Figure 5.28), which recognizes that the intention is to show enthusiasm, and thus the semantic value of the gesture that should accompany the speech has to be *Enthusiastic*. The model selects the appropriate gesture, and synchronizes it with the beginning of the speech.

Mini then moves on to explaining the game's rules to the user. During this while process, the communicative intention of the robot is *Explain*, which coincides with the semantic value of the gestures that have to be used. Mini tells the user that he will be asked a series of questions about History, and for each question he will be presented with four options.

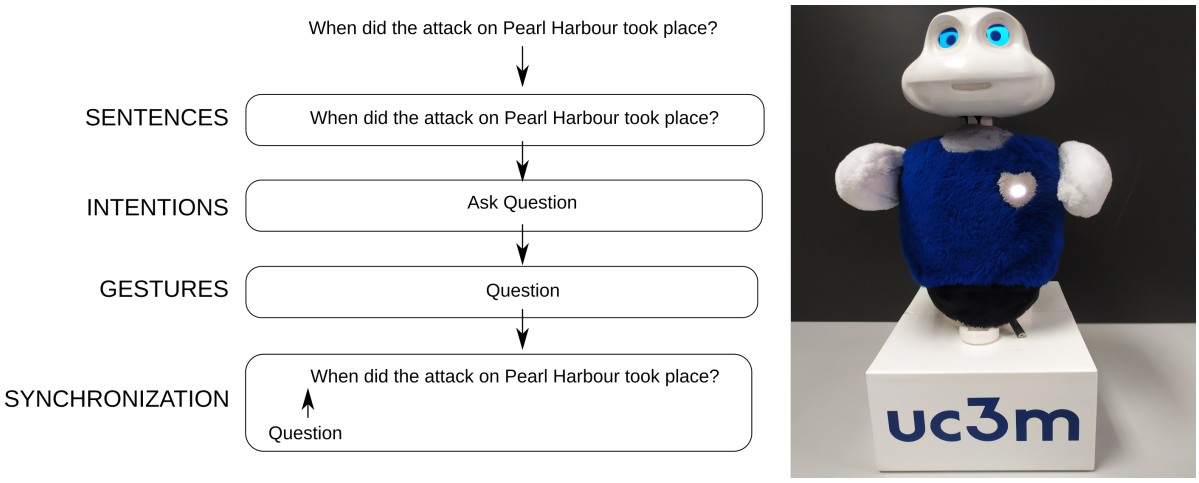


Figure 5.29: Process followed by the co-speech gesture prediction module follows for creating a multimodal expression for asking the first question of the quiz game. The image on the right shows Mini in the process of performing the selected expression.

After all the explanations have been completed, Mini asks the first question: *"When did the attack on Pearl Harbour took place?"*. Again, the question is also passed through the co-speech gesture prediction module (see Figure 5.29). Among the three communicative intentions included in Table 5.3 that could imply the use of a question, the one that fits this particular situation is *Ask Question*, as the goal is not to retrieve personal information or an opinion from the user. The model then ascertains that the proper semantic value of the gesture that has to accompany this utterance is *Question*, an expression where the robot takes a stance that tries to invite the user to provide an answer. Once again, after the question is uttered, Mini has to wait for the user to provide an answer. While waiting, the behaviours generated by the pulse-based liveliness are once again executed, as the robot is not performing any other expression. When the user provides the right answer to the

first question (1941), this stimulus triggers a change in Mini's affect state, making the *happiness* emotion to spike. As a reaction, the Emotion reaction module described in Section 4.5.5 requests the execution of the most appropriate expression for reacting to the stimulus that triggered the emotion. In this case, this expression congratulates the user for making the correct answer, with both verbal and non-verbal actions. Because this gesture is already multimodal, it is performed directly, without being sent to the co-speech gesture prediction module. After this congratulation has been uttered, Mini delivers some information about the correct answer, once again with the intention of explaining something, and thus the *Explain* semantic value is attached to these utterances. The rest of the questions in the game are performed in the same manner described here.

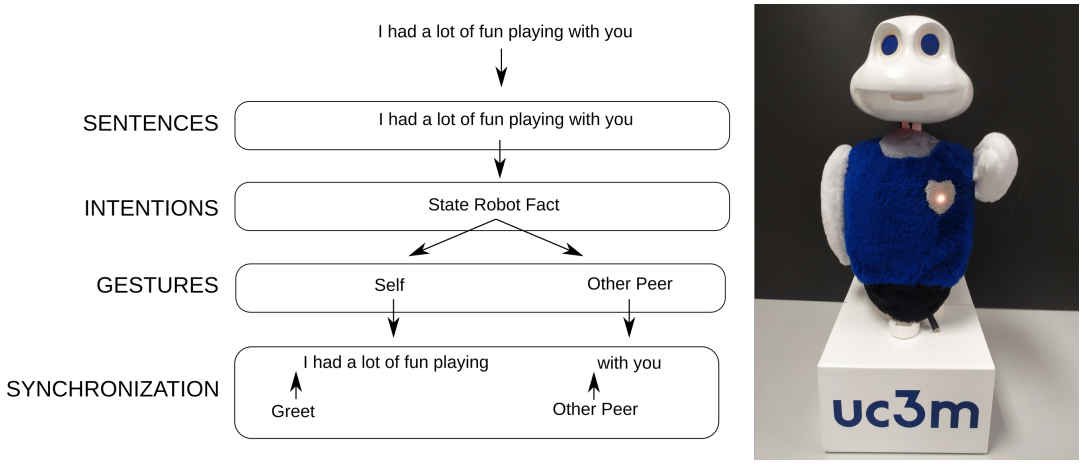


Figure 5.30: Process followed by the co-speech gesture prediction module follows for creating a multimodal expression for ending the interaction. The image on the right shows Mini in the process of performing the selected expression.

Once the game has been completed, Mini says to the user *"I had a lot of fun playing with you"*. The lack of an exclamation mark at the end of the sentence leads the model to infer that the communicative intention is to convey information about the robot, instead of showing enthusiasm (see Figure 5.30). Based on this intention and the utterance itself, the gesture prediction model labels the sentence with two gesture labels: *Self* and *Other Peer*, and synchronizes the appropriate gestures. With the one tied to the label *Self*, Mini tries to convey the idea of pointing at itself. Due to Mini's lack of elbows, this had to be expressed by lowering the head and the gaze so the robot seems to be looking at itself. With the gesture tied to the *Other Peer* label, Mini points at the user. The first expression is connected to the beginning of the utterance, while the other is tied to the beginning of the chunk *with you*. Finally, Mini uses a predefined expression to say goodbye to the user, and goes back to the standby state, completing the interaction.

5.5 Conclusions

Endowing a robot with a liveliness appearance is one of the factors that can lead to a human considering it a viable interaction partner, and easing the creation of bonds between them. This is essential for a social robot that will be integrated in a human environment. There are several methods that can be used to convey this liveliness appearance through the performance of non-verbal expressions. Examples of this are displaying motions that are not caused by an external force (for example, an user pushing the robot), suggesting the existence of a conscience behind the movement, or displaying human-like facial expressions. But when these non-verbal expressions have to be performed alongside speech, it is not enough that they are able to convey a liveliness appearance, but it is also important that they match the communicative goal of the verbal message. This chapter has presented two different methods for generating behaviours oriented to endow a social robot with a lively appearance.

5.5.1 Contributions and achievements

This chapter has aimed at developing strategies for enhancing the animacy of a social robot through the use of non-verbal behaviours. This led to the two main contributions presented in this chapter. First, a method for generating unimodal actions for a social robot has been proposed. This method represents the state of the robot as a sinusoidal signal with variable frequency and amplitude. This signal was designed with the idea of mimicking a heartbeat. The amplitude and frequency of the signal can be altered based on the internal state of the robot. In the current version of the pulse-based liveliness module, the affect state of the robot is considered to alter the values of the signal, along with an extra state where the robot simulates being asleep, and thus the generation of behaviours have to be stopped. At a certain rate, the pulse-based liveliness module samples the signal, and generates the appropriate behaviours. This is done by selecting randomly one of the available templates for each interface, and parametrize it using the sample taken from the signal.

The second contribution is the development of a co-speech gesture prediction method that can be used to enhance messages that only include verbal information. The proposed pipeline frames this task as a labelling problem, where the model has to generate an appropriate sequence of labels given the input information. The process has been divided into two steps: (i) using the words in the utterance and their Part-of-Speech tags to predict a sequence of labels indicating the communicative intention/s of the utterance; and (ii) combine this communicative intention/s with the inputs in step 1 to predict a sequence of labels indicating the semantic value of the gesture/s that should

accompany the speech. Each step is performed by a separate model, both designed as a combination of Long-Short Term Memory networks for encoding the inputs (one network for each type of input: words, Part-of-Speech tags, and intention labels) and a Conditional Random Field for obtaining the final sequence of output labels. A filter is added after each model to correct any possible prediction error that can affect the next steps in the process.

The last contribution that has been presented in this chapter is the development of a method for synchronizing verbal and non-verbal messages based on a set of rules. Developers can assign individual rules to each possible semantic value label, in order to define individual synchronization strategies for each type of gesture in the system. For example, an expression where the robot says no by shaking the head can be attached to the apparition of the actual word *no* in the utterance, while gestures used to greet the user could be attached directly to the beginning of the speech chunk. Each rule allows to define which feature should be used for the synchronization process (a position in the chunk, or the appearance of a given word or Part-of-Speech label), a particular value for this feature (if it has to be synchronized with the beginning, middle, or end of the chunk, or the actual word or Part-of-Speech label that has to appear), and other information required for the synchronization process (if the gesture has to be performed only once or if the entire chunk should be filled with as many gestures as possible, if an offset should be introduced...). Multiple rules can be defined for each possible semantic value, and the system will try to evaluate them in the order in which they are defined. Based on this rules and on a mapping that connects the semantic values predicted by the model with the expressions stored in the library, this synchronization method computes the starting point in seconds for each non-verbal behaviour, from the moment the verbal message is started to be uttered.

Both the pulse-based liveliness module and the co-speech gesture prediction module have been evaluated with objective tests, and a case of use has been presented to demonstrate how both methods are integrated in the HRI architecture of our robotic platforms. The results obtained for the pulse-based method show that the performance of this approach allows for a successful integration in our robotic platforms. However, while the proposed gesture prediction model has proven to be able to select the correct type of gestures in the majority of situations, and achieved a proper synchronization of the verbal and non-verbal modalities, the module showed that it might require more processing power than the one provided by our robots, in case we want to avoid excessive delays in the interaction.

5.5.2 Achievement of the proposed goals

The main objective of endowing the robot with a liveliness appearance was divided into two main objectives: (i) implementing a method for generating unimodal actions to be conveyed through each of the robot's communicative channels; and (ii) the development of a co-speech prediction method for enhancing the communication capabilities of the robot with the addition of non-verbal behaviours that match the robot's verbal message. Both objectives have been achieved successfully with the development of the pulse-based liveliness and the co-speech gesture prediction methods. These objectives have been divided in a series of subgoals:

- The first subgoal stated that the actions created by the pulse-based liveliness module have to be adapted to the internal state of the robot. This objective was successfully accomplished with the integration of the signal representing the robot's pulse. The actions created by the Interface modules are configured using samples extracted from this signal. The robot's pulse is defined by two parameters: the amplitude of the sinusoid and the pulse's frequency. At a certain rate, the Signal module checks the internal state of the robot, and alters the values for both parameters, if required. Thus, by altering the shape of the signal, the appearance of the actions generated by the Interface modules will also be altered. The effect that the robot's state will have depends on the type of action (for example, in motions, the final position in the trajectory is altered, making the movement longer or shorter).

- The second subgoal requested that the non-verbal expressions that have to be used to enhance the verbal message are selected based on the intention of the utterance, so the expressions help to achieve the communicative goal stated by the verbal message. This was achieved by dividing the prediction process into two different steps: (i) the first model receives the utterance and predicts a sequence of labels that state the communicative intentions of the different parts of the utterance; and (ii) the second model receives the combination of utterance and the output from the first model and predicts a sequence of semantic values for the gestures that have to be added to the verbal message. Both models have the same internal structure, a combination of LSTM encoders and CRF for predicting the labels. The difference is that the second model adds an extra encoder for the intention labels.

- The third subgoal required the creation of proper datasets for training the models requested in the second objective. First, a dataset was created for training the gesture prediction model, in order to test if the performance was satisfactory. The utterances for this dataset were extracted from the Cornell Movie Dialogs Corpus, and then labelled manually with both the intention labels and the semantic values for the gestures. Once the gesture prediction model was trained

with this dataset, and the performance was evaluated, the intention prediction model was developed, and the second dataset was created by just removing the information about the semantic values from the instances of the first dataset.

- The fourth subgoal involved the development of a synchronization method able to pair the utterance with non-verbal expressions based on the semantic values predicted by the machine learning pipeline. This objective was successfully implemented with a combination of a mapping that establishes the relationship between the semantic values predicted by the model and the expressions in the library, and a rule-based synchronization method. Another requisite was that the system should be able to manage an undetermined amount of expressions. In order to do this, the utterance is split in multiple speech chunks, based on the semantic values. Then, the synchronization process is performed for each chunk individually. This allows to synchronize as many expressions as required.

- The previous objective was extended with the fifth subgoal, which stated that the gestures added to the verbal message should be loaded from a gesture library, and that the synchronization process should be easy to adjust, so it can be adapted to the expressions available. The first part of the goal was achieved with the implementation of a mapping between the semantic values and the gestures in the library. A configuration file contains a dictionary where developers can specify a list of expressions that correspond to each possible semantic value. The length of the expressions has to be also specified, so the synchronization method can evaluate if a particular gesture will fit in a speech chunk. The second part of the objective was achieved with the implementation of the synchronization rules. These rules are easy to craft, and multiple rules can be defined for each semantic value, which gives developers a more precise control over the behaviour of the synchronization method.

- The sixth subgoal required that the prediction and execution of expressions should be detached in the co-speech gesture prediction model. This would help to integrate the proposed method in robotic platforms with different output interfaces. This goal can also be marked as completed. The proposed method establishes a clear division between the prediction of gestures, the synchronization of the verbal and non-verbal messages, and the generation of the final expression that has to be executed. A machine learning-based pipeline receives the next utterance of the robot, and predicts the semantic value of the gesture/s that should be added. This is then passed to the synchronization method, which is in charge of selecting the proper gestures from the library and returning the starting point (in seconds) for each gesture. Both processes are mostly platform-independent (they do require an expressiveness architecture that uses predefined expressions, and that allows to add a delay to the start of a gesture). The last step is

the only one that is completely platform-dependent, as the utterance and the gestures selected have to be formatted in a request that can be processed and executed by the expressiveness architecture.

- The last subgoal stated that the proposed liveliness methods should be integrated in the robot. This implies that their performance allows for a fluid and natural interaction. Based on the results of the objective tests presented in Section 5.4.1, this objective has been partially completed. While the pulse-based liveliness method was successfully installed in the robotic platforms used in this dissertation, the co-speech gesture prediction module required an excessive amount of resources, and introduced a delay in communication that could lead to unnatural interactions if the utterance that is passed to the method is too large.

5.5.3 Limitations of the system and future lines of work

During the development and evaluation of the proposed methods for enhancing the animacy of a social robot, several weak points and potential future lines of work were identified, in order to improve the performance of both the pulse-based liveliness and co-speech gesture prediction methods:

- The pulse-based liveliness method uses a sinusoidal signal to represent the pulse of the robot. This signal can be adapted to the internal state of the robot, which in turn results in a change on how the different unimodal actions are generated. While this approach allowed to change the non-verbal behaviours so they tried to represent the robot's state, it presents one main limitation: it only changes how an action is parametrized, but it has no effect over which type of action is performed. For example, while happiness can increase the amplitude of the robot's motions, and sadness can cause the opposite effect, the type of motion is the same under both states (moving the arms back and forth, for example). Future works should try to extend the effect that the robot's internal state has over the generation of behaviours. A possible solution would be to develop different action templates for each possible state, and pass the state of the robot alongside the sample extracted from the pulse signal to the Interface modules.

- While it is important that the behaviours performed by the robot are adapted to its internal state, this is not the only factor that could play a role on how the unimodal actions created by the pulse-based liveliness method are generated. Other circumstantial factors could be taken into account to improve the robot's appearance. For example, the behaviours generated could be different if the robot is interacting with a user, or it is alone. This way, if the robot is in the middle of an interaction, the motions generated could be oriented to follow the other peer, or show the appearance of being paying attention. In order to extend the proposed Liveliness

module with this feature, three steps would have to be conducted: (i) identify the context factors that should have an effect over the robot's behaviours; (ii) identify how each new factor should affect each of the robot's communicative interfaces; and (iii) modify the Signal and the Interface modules so they take into account these factors.

- The objective evaluation of the proposed co-speech gesture prediction method showed that the required amount of resources to run the model was too high for the robotic platforms in which the model was tested. Also, the delay introduced in the interactions by the prediction and synchronization of non-verbal expressions was slightly too high when a short utterance was being processed, and this delay grows with the length of the sentence. Thus, and alternative solution for running the proposed model should be found. Two possibilities have been identified so far, although other options could be considered. The first possible solution would be increasing the robot's processing power with hardware designed with machine learning in mind. The second solution would be to extract the machine learning pipeline from the robot and run it into an external machine that is prepared for this type of approaches. The robot would then communicate with this machine, sending the robot's utterance and receiving the predicted sequence of semantic values.

- In order to synchronize the verbal and non-verbal modalities, one of the things that the synchronization model needs to know is the length of the speech. This is important for computing the starting point for the non-verbal expressions, and also for identifying if a particular gesture can be performed in the time it takes to utter the verbal message. The current version of the system measures the length of the utterance in base to the number of characters in the text that will be sent to the TTS. Sentences with different lengths were sent to the TTS, their duration was measured by hand, and a relation between the number of characters and the length of the utterance was obtained. While this approach has proven to work acceptably well, it is still an approximation, as it does not take into account other factors that will change the time required for uttering a sentence. For example, the speech patterns used for a question could be different to those used for a statement, and thus the estimated relation between the number of characters and the length in seconds might be also different. Also, the expressiveness architecture used in our robots allows developers to specify the prosody rate for a particular utterance, which is also not contemplated in the proposed synchronization method. Thus, a new approach should be developed for obtaining the length of an utterance. Because one can not rely on the TTS providing this information, the new method should still aim at computing this length theoretically.

- When the synchronization method has to load expressions from the library to pair them with the verbal message, the algorithm checks the mapping specified in the configuration file, which indicates not only which gestures correspond to each semantic value, but also their duration (in seconds). Based on the length of a particular utterance chunk, and the semantic values with which it has been labelled, the synchronization method extracts all the gestures that could be performed in the time required to utter that particular chunk, and then selects one of them randomly. This could lead to situations where no gestures can be selected for a given chunk, or where the length of the chunk is much larger than the duration of any gesture, which could lead to the robot remaining still for too long. While the idea is to have gestures with different lengths for each possible semantic value, and the synchronization rules allow to mitigate the second problem by performing the same gesture multiple times to fill the entire duration of the chunk, the system could benefit from the implementation of a method that can modify the non-verbal expressions to alter their length. The idea is to allow the synchronization method to extend or shorten the duration of an expression in order to adapt it to the length of the utterance chunk. This is not a trivial process, as the modification of the expression's duration should be performed in a way that maintains the integrity of the message being transmitted by the expression.

- In the current version of the system, a limited amount of gestures have been developed for each possible semantic value. While this was enough for testing the performance of the proposed approach, more expressions should be included in order to increase the variability of the robot's expressiveness, to avoid an excessive repetition of gestures. Thus, the library of expressions should be extended with more examples for each possible semantic value.

CHAPTER 6

Conclusion

Robots are starting to become a part of our daily lives. While the advances on robotic technology are starting to make robots more cost effective than human workers [1], demographic changes predicted for the near future are creating problems that could be solved with the integration of robots in different sectors of our society. However, in the case of the services sector, it is necessary that these robots are able to interact with humans in a way that abides by the social rules enforced in the domains in which the robot will be inserted, and meet the expectations that the humans in said domains have for human-human interactions. This dissertation was born from the need to endow social robots with these capabilities.

6.1 Contributions and achievements

The main goal of this dissertation was to endow a social robot with the necessary communicative abilities in order to provide an interaction that feels satisfactory to a human user. With this objective in mind, this dissertation has made contributions to three main topics: (i) dialogue management, (ii) expressiveness management, (iii) and animacy display.

Dialogue management is one of the key problems in the field of Social Robotics. Creating frameworks that allow robots to imitate the way that humans communicate between them is essential for developing robots that can be integrated in real environments and replace human workers in areas that involve contact with users. Along the years there has been an extensive body of research in this area. The development of artificial intelligence and deep learning has brought a big impulse

forward in the creation of systems that can interact like a real human, although a big part of the work developed focuses on speech-based interactions. Multimodal communication is essential for robotic platforms, and should be the direction in which the research on dialogue management for robots tends to.

Chapter 3 of this manuscript has presented the proposed approach to dialogue modelling and management for social robots. The work conducted in this area has led to two main contributions. First, a dialogue model has been proposed for representing one-to-one interactions between a robot and a human. Under this model, interactions are divided in basic units called *Communicative Acts*, or CAs. These CAs have been defined based on two dimensions of the dialogue: (i) which speaker is holding the initiative in the interaction; and (ii) what is the communicative goal of the speaker with the initiative. Although speakers can have a large amount of possible communicative goals (e.g. make a request, ask a question, greet the other peer...), this dissertation considers that all of these goals are variations of one of two final goals: give or retrieve information. Thus, the basic CAs (or BCAs) have been designed to complete one of these two communicative objectives. Because these CAs are highly parametrizable, developers can adapt them to achieve more specific objectives. The control over dialogues is this divided in two levels, where the robot's applications create the flow of the interaction as a combination of CAs based on task-related information, while the HRI Manager controls interaction-specific aspects of the dialogue. Finally, CAs can be combined into more complex structures, called *Complex Communicative Acts* (also known as CCAs). In turn, these CCAs can be also combined in an hierarchical manner with either BCAs or other CCAs.

The second main contribution that Chapter 3 presented was the development of the HRI Manager, the software architecture required for implementing human-robot interactions using the CAs. On top of providing the methods required for loading, configuring and executing the CAs, it also provides a series of functionalities that are required in the majority of interactions. While the CAs implement a series of mechanisms for error handling during communication, the HRI Manager provides a priority-based conflict management mechanism for controlling when to start, pause, or cancel dialogues.

While having the tools for managing interactions in a way that feels natural to the user is a cornerstone in the development of a social robot, it also requires that the actions selected by the dialogue management approach are modelled as expressions that seem human-like. This involves the use of multiple communication modes (speech, motions, gaze, etc...), which have to be appropriately coordinated between them. Researchers have faced this problem using a variety of techniques, which

go from handcrafting each action individually to using machine learning models to generate the expressions.

Chapter 4 focused on the design and development of an expressiveness architecture for a social robot. Here, three main contributions can be highlighted. The first contribution has been the proposal of a model for describing multimodal expressions for a social robot as state machine-like structures. This solution allows to consider all the robot's output communication channels at the same time during the design of gestures, instead of focusing on each interface individually and then combining all the modalities. The proposed approach allows to connect the execution of actions to specific moments in time (e.g. 3 seconds after the execution of the gesture starts), or to the conclusion of previous actions (e.g. raise the left arm when the right one has finished its motion). Under the proposed expressiveness architecture, expressions are based on FlexBE, a framework for developing behaviours for robots. This allows gestures to be created using a graphic interface. Also, in order to increase the flexibility of the system, a template was developed for creating multimodal expressions from a list of individual actions.

The second contribution related to the expressiveness of a social robot has been the development of the Expression Manager, the module in charge of controlling the robot's expressiveness. Its tasks involve managing requests for executing gestures and schedule them (Expression Scheduler), and loading them from the expression library and ensuring that their execution is completed satisfactory (Expression Executor). The Expression Manager also provides serves as an interface between the software architecture and the robot's actuator control modules (e.g. the ETTS, the drivers for the motors...) through the Interface Players. The last contribution of the work that has been presented in Chapter 4 is the addition of three of mechanisms for adapting the expressiveness to the robot's internal state: (i) allowing developers to replace individual actions inside a predefined multimodal expression (e.g. change the utterance of a greeting gesture); (ii) using two control parameters (speed and amplitude) to change the overall appearance of an expression; and (iii) using modulation profiles to specify the effect that each possible internal state has over the different communication interfaces (e.g. happiness raises the pitch of the voice, sadness lowers the amplitude of the motions).

There is a conclusion that can be extracted from the work developed in Chapter 4. According to research [114], human communication can be divided in two components: symbolic and spontaneous. While the former has the objective of conveying messages, the later involves the display of emotional and motivational states. From this, it can be deduced that an expressiveness approach for robotics cannot be devised exclusively with a functional mentality (the only goal of the architecture

cannot be conveying communicative messages with a defined goal). Instead, is important to merge both dimensions of communication, so the robot is able to include its own internal state in the messages being conveyed to achieve communicative goals. The proposed approach was designed to manage both aspects of communication, in order to be able to convey complex multimodal messages while abiding by the rules that control interactions between humans (particularly, the temporal constrains involved in all dialogues), while at the same time providing the tools required to transform these multimodal expressions so they are able to convey the internal state of the robot while still being able to complete their communicative goal.

The last topic discussed in this dissertation is the importance that displaying a liveliness appearance has for a social robot, in order to be recognized as a suitable interaction partner by humans. While conveying this appearance is a complex task that involves aspects like the external design of the robot, or its cognitive abilities (the capacity for learning new behaviours, for example), research reviewed in this chapter has shown that an appropriate use and combination of verbal and non-verbal expressions can help to enhance the robot's animacy. Two different situations have been identified: (i) the robot performs expressions that have no communicative purpose (do not try to convey any particular message) to enhance its animacy; and (ii) the expressions performed by the robot try to enhance its animacy while trying to complete a particular communicative goal. While the former could be connected to the spontaneous dimension of communication identified by Buck and VanLear [114], the latter combines both dimensions (symbolic and spontaneous).

This thesis has focused on how non-verbal behaviours can be used to increase the robot's animacy. The work conducted led to two different approaches. The first method focuses on creating liveliness behaviours for social robots based on a sinusoidal signal that represents the robot's heartbeat. The internal state of the robot is used to adapt this signal, which is then sampled at given intervals. The samples extracted from the signal are then used by a series of modules to generate unimodal actions. For each communicative interface, a series of templates depicting different possible actions (e.g. different types of arm motions, or different gaze patterns) have been designed. When an action has to be created, each Interface module in the pulse-based liveliness method selects one of this templates and then parametrizes it based on the signal. The second approach seeks to enhance the communicative abilities of the robot by selecting the non-verbal behaviours that have to accompany the speech of the robot, based on the content of the utterances and the communicative intentions of the verbal component. This task was solved as a labelling problem, where a deep learning model assigns labels to each word in the utterance indicating the semantic value of the gesture that should be selected. The proposed model first predicts the communicative

intention/s of the utterance based on the words and the Part-of-Speech labels assigned to each word in the utterance. Then, the sequence of intention labels is combined with the words and Part-of-Speech labels and used to predict the list of semantic values for the gestures. Both the intention and gesture prediction are done using a model that combines an LSTM-based encoder and Conditional Random Field for predicting the sequence of labels. The last contribution in this area is the development of a synchronization method that receives the utterance and the predicted sequence of semantic values, selects the most appropriate expressions from the gesture library, connects them to the appropriate points of the utterance, and creates a single multimodal expression that contains both the verbal and non-verbal components. This method relies on a set of rules that are handcrafted by the developers for each possible semantic value. Each rule defines the type and value of the utterance's feature that should be used to find the point where the gesture should be connected (the appearance of a given word, Part-of-Speech label, or a predetermined point in the chunk), as well as other extra information required for completing the process (e.g. if a utterance chunk should be filled with as many gestures as possible, or use an single gesture). For each semantic value, developers can create multiple rules that the system will evaluate sequentially.

The work that has been presented in Chapter 5 led to several conclusions. In the first place, endowing a robot with a liveliness appearance through the use of non-verbal expressions is not a task that can be conducted independently, instead is deeply connected to the robot's expressiveness. Any method that tries to convey animacy with the performance of specific expressions should take into account the communicative goals that the robot might have at any given moment, so these behaviours do not hinder the interactions conducted between the robot and the users. Second, although some researchers defend that animacy is one of the dimensions that expressions have, it is not clear that the same behaviours can be used under every circumstance to display animacy. On the contrary, it seems beneficial to adapt these gestures to different factors, both related to the internal state of the robot and the context of the interactions, so the behaviour of the robot is perceived as natural. Lastly, while non-functional behaviours for conveying animacy might not be subject to temporal restrictions (they can be performed at random moments in time, and with a variable frequency), when they are performed simultaneously with functional behaviours, then they need to abide by the same temporal rules that control any interaction. This means that the selection/generation of the non-functional expressions has to be done at a speed that allows for a natural interaction, and that a proper synchronization between the functional and non-functional behaviours has to be achieved.

All the modules presented above have been tested through both objective and subjective evaluations to ensure that they meet the requirements that human-robot interactions impose on

the robot (time required to convey information to the user, the ability to overcome unexpected situations that arise during the dialogue, etc...). Also, because the proposed architecture has to be integrated in robotic platforms, the amount of resources that each module requires to work is also a key metric to validate the work performed. Thus, the objective evaluation was conducted for all the modules presented above, measuring the use of RAM and CPU, and also the response time. The results show that the HRI Manager, the Expression Manager, and the pulse-based liveliness approach are all able to operate at a speed that fulfils the requirements for human communication, and with a performance that allows both methods to be integrated in a robotic platform with limited resources. However, although the proposed co-speech gesture prediction method is able to successfully select the most appropriate non-verbal behaviours given an utterance, it requires an excessive amount of resources, and could introduce excessively high delays in the communication process if the utterances are too large.

Regarding the subjective evaluation, cases of use were presented for both liveliness methods and the HRI Manager. For the Expression Manager, the evaluation focused on the modulation strategies proposed, through two video-based experiments. The first one evaluated the effect that the parameter-based modulation has over how users perceive the robot, while the second experiment studied the effect that the display of affect states has over the users' perception, using the profile-based modulation approach. The case of use presented for the HRI Manager showed that the robot was able to conduct interactions with patients in a daycare centre using the CAs to build these interactions. The majority of interactions were conducted without issues, and in those where errors arise, the mechanisms integrated in the CAs were able to handle these situations successfully. The case of use showcases some key situations observed during the trials, and that can be used as an example of the capabilities of the proposed approach to dialogue management. Regarding the liveliness methods, the case of use describes a common application of the robotic platform used in this thesis (a quiz game where the robot asks multiple choice questions to the user), and uses this example to demonstrate how the proposed liveliness approaches are integrated in the robot's software architecture. Finally, the experiments conducted with the Expression Manager showed that a correct modulation of the robot's expressiveness might improve the perception that users have of the robot's capabilities. While the parameter-based modulation succeeded in improving the participants' opinions, the results of the experiment conducted using the profile-based modulation were not that positive. Further tests should be conducted to have a clearer understanding of the reasons behind these bad results.

Overall, the evaluations conducted showed that the proposed Human-Robot Interaction architecture can be effectively integrated in a robotic platform, and allow a social robot to interact with humans in a way that feels natural. Thus, the main goal of this dissertation is completed.

6.2 Achievement of the proposed main goals

In Chapter 1 of this dissertation, the main objective of endowing a robot with a series of communicative abilities that allow it to maintain interactions that feel natural to the user was decomposed in a series of subgoals that could be used to measure the degree of success obtained with the work developed. While the objectives presented in Sections 3.1.1, 4.1.3, and 5.1.1 were tied to the three parts in which this thesis can be divided, the subgoals presented in Section 1.3 were designed from a more general point of view. In this section, the degree of each subgoal will be analysed in depth:

1. The first subgoal required that the proposed Human-Robot Interaction architecture developed had to be simple to use. The idea is that interactions should be easy to create, in order to simplify the addition of new applications. This goal was tackled through the implementation of two modules: the HRI Manager and the Expression Manager. With the CA-based approach to dialogue management, developers can create interactions by combining and parametrizing basic interaction units. The configuration of these units is done by filling a standardized request that will be sent to the HRI Manager core. The use of an standard message reduces the learning curve for new developers, and also the time required for creating interactions. Also, the HRI Manager takes care of a series of tasks that are common to all interactions, once again reducing the workload of the developers. The proposed approach to expressiveness management, on the other hand, tries to simplify the process of creating expressions by allowing developers to use a graphic interface to create multimodal expressions by dragging blocks and connecting them to each other. Also, the Expression Manager allows developers to ignore completely the library of multimodal expressions and instead provide a list of actions that will be transformed into a gesture. This eliminates the need to create expressions for every communicative action that the robot has to perform, which could be time wasting if the expressions are not very complex. For example, the most common type of action requested is having the robot utter a sentence, and allowing developers to pass the utterance to the Expression Manager instead of creating a specific expression is significantly faster.

2. The second subgoal aimed at obtaining a modular system, where each part can be easily detached from each other, and replaced or updated. This goal was achieved with the creation of the HRI System, described in Chapter 2. This architecture divides the abilities required to create interactions between the robot and the users into four main elements: (i) the Perception Manager performs all the tasks involved with obtaining information from the robot's sensors; (ii) the Expression Manager controls all the expressive capabilities of the robot; (iii) the HRI Manager controls the interactions using the information provided by the Perception Manager and requesting the execution of communicative actions to the Expression Manager; and (iv) the liveliness module creates expressions that can enhance the animacy of the robot. All the modules are connected using ROS, a framework for robotic applications that provide standard communication channels between modules. With this, every element in the software architecture acts as a black box for the rest of modules. Thus, adding or replacing parts of the architecture only requires that the inputs and outputs of the new module match the ones expected by the rest of the architecture. On top of this, the modules presented in this thesis has also been developed as a combination of blocks (for example, the Expression Manger is divided in the Expression Scheduler, the Expression Executor, and the Interface Players).

3. The third subobjective required that the proposed architecture should be easy to integrate in multiple robotic platforms and be application independent. Due to the fact that the proposed HRI System was developed using ROS, a requirement for integrating this system in new platforms is that they use this framework. However, this is not an important drawback, as ROS is a popular solution amongst roboticists. Independence from the robot's applications was achieved with the division of the control over interactions in two levels, where the applications hold all the knowledge that is task-specific, while the HRI System focuses in those aspects that are dialogue-independent. The CAs can be adapted and parametrized to be used in any application (as far as we know), and also be combined to create the dialogue structures required for achieving any specific communicative goal. Thus, in order to adapt the proposed architecture to a new platform involves three main steps (granted that the condition of that platform's architecture being designed in ROS is met). First, the HRI Manager core expects that the input information coming from the robot's sensors is formatted in a specific way, imposed by the Perception Manager. Thus, an extra layer would have to be added to the HRI Manager core to convert the inputs provided by the new software architecture to the format that the CAs require. Second, the Interface Players are the only elements in the HRI System that communicate with the robot's output interfaces. When integrating the system in a new platform, the Players have to be adapted (or new ones have to be developed) in order for them to create commands that the robot's output modules can understand. The last change involves

modifications not to the proposed HRI System, but to the applications of the new platform. They should be adapted so they use the CAs for creating interactions. While the two first modifications are not very complex, the last one depends on how the applications are designed. This leads me to consider this goal to be achieved.

4. The fourth subgoal required that the proposed system had to be flexible and adaptable to different users and the particularities on an interaction, in order to make communication as natural as possible. This feature was added through both the HRI Manager and the Expression Manager. CAs are highly parametrizable in order to achieve different communicative objectives, and can be combined to create more complex structures. In the expressiveness management architecture, three methods were implemented to modify the appearance of the robot's expressions in runtime, in order to adapt them to different aspects of the interaction and the state of the robot. Also, one of this methods allow to replace actions in a predetermined expression in order to adapt it to a new situation. Finally, developers can use keywords when developing the verbal messages that the robot will convey. These keywords are then replaced by the ETTS Player with information extracted from the context of the interaction (e.g. the name of the user, the location where the interaction is taking place, the moment of the day...). With this, this goal can be considered as successfully completed.

5. The fifth subgoal was related to the evaluation of the proposed system. It required that this evaluation was conducted among a population not restricted to users that belong to the field of application of the robotic platform used in the evaluation (older adults). While the case of use presented in Chapter 3 depicted a series of tests that were conducted with older adults at a day care centre, the experiments presented in Chapter 4 were conducted with participants of different backgrounds. Although these experiments had the goal of evaluating the propsoed modulation approaches, the appearance of the robot under all the conditions in both experiments is the product not only of the modulation strategies, but also of the result of modelling interactions as a combination of CAs in the HRI Manager and the expressiveness generated by the Expression Manager. Thus, while the differences between conditions are attributed to the modulation strategies, the ratings that the participants gave to the robot's *warmth*, *competence*, and *discomfort* represent the performance of the entire software architecture. Due to this, this goal was considered to be completed.

6. The sixth subgoal focused on the reaction time of the proposed system. With the HRI System, the robot should be able to abide by the temporal constraints that are involved in human communication. In Chapter 3 it was determined that a reasonable threshold for the time required to convey information to the user could be set around 1 second. Chapter 4 extended

this explanation with a study of the reaction time that humans display when reacting to certain stimuli in the environment. In order to ensure that all the proposed modules are able to perform under the thresholds considered, an objective evaluation was conducted to measure their reaction times. The results show that both the HRI Manager and the Expression Manager can complete their tasks at a speed that ensures that the threshold for a natural interaction can be achieved without putting too much pressure on the parts of the architecture that are outside the scope of this thesis (for example, the time the applications require to advance the dialogue). However, the evaluation of the co-speech gesture prediction method showed that the delay introduced by this module can be excessive depending on the length of the robot's utterances. Thus, this objective can be considered as partially achieved.

7. The final subgoal stated that the architecture had to be integrated and running in a robotic platform. This means that the hardware resources will be limited, and have to be shared between all the modules in the software architecture. Thus, the proposed architecture not only has to abide by the temporal constraints of human communication, but has to do so without hoarding an excessive amount of resources. Due to the importance of ensuring this, the objective evaluations conducted in Chapters 3.1.1, 4.1.3, and 5.1.1 included also an evaluation of the RAM and processing power used by each module. Similar to the results observed for the reaction times, the only module that showed an excessive usage of resources was the co-speech gesture prediction method. This was not surprising, as the robotic platforms used in this thesis lack hardware oriented to running deep learning model (e.g. a graphics card). Nevertheless, this leads me to consider this goal as being only partially achieved.

6.3 Final remarks

Robots that can live and cooperate with humans is one of the aspects that a lot of people imagine when they think about what the future of society involves, as shown by a large amount of movies, books, videogames, etc... The development of the work that has been presented in this dissertation has been an exciting challenge that has given me the opportunity of getting involved on one of the key tasks that separate us from that future: giving robots the ability to interact as humans.

Among the research topics present in this dissertation, dialogue management is the one that attracts more attention nowadays. With the proliferation of smartphones and home assistants, the way in which we interact with technology has shifted in recent years towards more natural ways of communication. While a big portion of the work conducted in this area has focused on giving people the ability to use natural language to control their electronic devices, there is also a

high interest on integrating multimodal sources of communication in this process, specially when we talk about robotics. One of the lessons that I have learned during my work in this area is that giving robots advanced communicative skills can enhance how users perceive them, but at the same time can complicate the process of increasing the range of tasks that robots can perform. With the work presented in Chapter 3, the key concept that I had in mind from the first moment was the importance of finding a way to balance the naturalness of the robot's communicative skills with the scalability of the set of tasks that the robot can complete. This is what has driven me to focus on developing a system that is highly modular and parametrizable, so it allows developers to design their applications as they might see fit. This can be seen not only on the work developed on dialogue management, but also on the expressiveness architecture developed.

To the best of my knowledge, the majority of works related to animacy in robotics are studies that evaluate how different actions and environmental conditions affect to an entity's animacy, while a minority of works have tried to integrate in a robot methods for generating behaviours with the only goal of increasing the robot's animacy. Due to the importance that displaying a liveliness appearance seems to have for social robots, it is crucial to use all the knowledge that years of studies on animacy and robotics have into implementing appropriate strategies that can be integrated in robotic platforms. I hope that Chapter 5 of this dissertation has contributed to this area in a significant manner.

Overall, this thesis was born with the idea of setting the foundations for endowing a robot with all the required abilities for conducting interactions with users in a natural way. While the final architecture has proven to be capable to manage human-robot interactions appropriately, the feedback that we have received from users that have tested our robots has convinced me that our system still has room for improvement, something that I am looking forward to.

APPENDIX A

Theoretical base of the co-speech gesture prediction method

This appendix introduces the theoretical foundations of the algorithms used during the development of the co-speech gesture prediction method: Long-Short Term Memory neural networks and Conditional Random Fields

A.1 Long Short-Term Memory Neural Networks

Recurrent Neural Networks (from now on RNN) are a type of neural networks that have been endowed with feedback connections in order to learn patterns that are sequential or vary through time [215]. They are derivations of feed-forward neural networks with the ability to use internal memory to process input sequences with variable length. The typical structure of a RNN includes an input layer, one or more hidden layers, and an output layer. The difference with traditional feed-forward networks is on how the hidden states in the network are connected. In these hidden layers, the outputs in previous steps are introduced as inputs for the current time point. Thus, the hidden state of the network for a given timestamp can be expressed as:

$$h_t = f(W_h * h_{t-1}, W_x * x_t) \tag{A.1}$$

Where h_t is the output of the hidden layer at the current time point, f is the fixed function that has trainable weights, h_{t-1} is the output of the hidden layer at the previous time point, x_t is the input to the network at the current time point, and W_h and W_x are the weights associated to the previous

state and the current input. Fields of application where this type of nets are used include examples like language generation, machine translation, speech or handwriting recognition, image recognition, etc... Among these applications, the one that it is of interest for this thesis is sequence prediction, in particular sequence-to-sequence problems, where an input sequence of steps are mapped into a different output sequence.

A key development in the appearance of RNN is the work proposed by Rumelhart et al. [216], where the back-propagation learning method was introduced. Under this approach, the weights of the connections between units in the networks are adjusted continuously to minimize the output error (the difference between the expected and the real output). Thanks to this adjustment, the hidden layers of the network will be able to represent different features of the task domain, while regularities in the task are represented by the connections between the units in the hidden layers. While weights in feed-forward networks can ben different for each node, in RNN they are shared by every node in each of the network's layers.

One of the taxonomies used for classifying RNNs has to do with the inputs and outputs of the network, and in particular, their size:

- **One-to-Many:** A single input is mapped into a sequence of outputs. For example, a system that receives an image as an input and generates a text description as an output.
- **Many-to-One:** A sequence is used as input to the network and mapped into a single output. An example of this is a sentiment classifier that receives a sentence as an input and assigns it a label indicating the sentiment (positive or negative)
- **Many-to-Many:** In this type of RNN, a sequence of inputs is mapped into a sequence of outputs. Inside this category, two different situations can be observed depending on if the inputs and outputs are synchronized. If they are, outputs are generated for each step in which an input is introduced in the network. If they are not, then there is a delay between the first step in which an input is sent to the network and the first where an output is obtained. This delay allows the network to take into account the entire (or part) of the input sequence for every element in the output sequence.

The back-propagation learning method used for training RNNs, which is commonly known as backpropagation-through-time (BPTT) presents two main problems that can appear when training a model: (i) vanishing gradient and (ii) exploding gradient. In this context, the gradient is the relation between the change rates of the model's error and the change in weight. In the first problem, a very

small gradient value can cause that the model stops learning, or at least takes a long time to do it. In the second situation, the training assigns an excessive importance to a particular weight, causing the gradient to increase continuously, tending to infinity. This can lead to the model crashing. Due to these problems, RNN are not able to learn long-term dependencies. Different techniques have been proposed throughout the years to counteract this two problems, like identity initialization or gradient clipping, among others. One of the methodologies proposed is the use of a particular implementation of RNNs: Long Short-Term Memory Networks.

Long Short-Term Memory neural networks (LSTM networks from now on) are a particular variation of RNN that is equipped with a series of units, known as LSTM units, that allow gradients to flow unchanged through the network, partially solving the vanishing gradient problem. The original idea for this type of networks was proposed by Hochreiter and Schmidhuber [217]. In this work, the central element of the network is the *memory cell*. The core of the unit is a central linear unit that has a fixed self-connection (this is known as the constant error carrousel, or CEC), which allows for constant error flow, and then is extended with input and output multiplicative units, also known as gates, which provide inputs to the memory cell. Overall, the CEC will store error signals that will not be changed, the input gate is used to decide if the information stored in the CEC has to be kept or overwritten, and the output gate is used to decide if other cells should be able to access the information stored in the CEC or not. According to [217], both gate types are not always necessary. Memory cells are grouped into blocks that share the same input and output gate, which makes it easier to store information.

In current implementations of LSTM units. they are modelled as the combination of a central cell and three different gates: (i) the forget gate is used to decide which elements of the information stored in the cell should be forgotten, (ii) the input gate controls which new information extracted from the current input is stored in the cell, and (iii) the output gate controls what parts of the memory should be output. All these gates are implemented as a combination of a sigmoid neural net layer and a multiplication operator. The inner working of an LSTM unit is as follows. First, the cell decides which information should be forgotten by generating a set of scale factors in between 0 and 1 for each element in the cell state. This is represented by Equation A.2:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{A.2}$$

Where f_t is the vector of scaling factors, W_f and b_f are the set of weights that control the forget process, h_{t-1} is the previous hidden state, and x_t is the current input. Next, the cell decides which new information should be stored in the cell. Two steps have to be completed: (i) selecting the elements

that will be overwritten and (ii) generate candidate values for these elements. This is represented in equations A.3 and A.4 respectively:

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{A.3}$$

$$\tilde{C}_t = \tanh(W_c * [h_{t-1}, x_t] + b_c) \tag{A.4}$$

In these equations, W_i, W_c, b_i, and b_c are the weights controlling the input selection and candidate generation processes, respectively. The final internal state of the cell will then be obtained through Equation A.5:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{A.5}$$

Where C_{t-1} is the previous internal state of the cell. The final step in the process is obtaining the output of the cell and the hidden state. First, the output gate selects the elements of the cell state that have to be output, and then the cell state is passed through a $tanh$ function to keep the values between 1 and -1, and multiplied by the vector generated by the output gate. This process is shown in Equations A.6 and A.7:

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{A.6}$$

$$h_t = o_t * \tanh(C_t) \tag{A.7}$$

While the process described is the one followed by the basic implementations of LSTMs, there are other variations that introduce new connections to the cells or that combine some of the gates presented above (for example, instead of removing information and adding new one through two independent processes, connect them in a way where information is only forgotten if something new has to be stored in its place). A commonly used variation is the Gated Recurrent Unit neural network, or GRU, presented by Cho et al. [218]. This approach combines the forget and input gates into a single one, and also merges the cell state with the hidden state.

While there are different methods that can be used to train a LSTM network, one of the most common ones is the algorithm known as Backpropagation Through Time [219]. This method starts by initializing the weights that control the three gates to the cell (input, output, and forget). Then, the input at a certain time point and both the previous hidden state and cell state are passed to the LSTM cell that is being trained. According to the equations presented above, the cell calculates both the output and the current cell state. After, using the derivative chain rule, the gradient is computed

through backpropagation through time. These gradients are computed with respect to the three gates, the current and previous cell state and also the weights for each gate. Finally, using these gradients, the weights associated to each of the gates are updated in order to reduce the error observed (the difference between the expected output and the one predicted by the network). Because the method considers all the timesteps in the input sequence during the computation process. it can be very slow. Thus, optimized versions of the algorithm were proposed to correct this problem.

A.2 Conditional Random Fields

Conditional Random Fields [220], or CRFs, are probabilistic models designed for segmenting and labelling a sequence of data. Traditionally, the approaches that have been widely used to solve this task are hidden Markov models and stochastic grammars. While this type of models have to assign a joint probability to the paired observation and label sequences, conditional models define the probability of a possible label sequence given an observation. This has the advantage of eliminating the need for modelling all possible observation sequences, and also can take into account arbitrary, non-independent features extracted from the observation sequence without having to account for their distribution. But some conditional models, like Maximum entropy Markov models still have a weakness that has to be addressed: the label bias problem. This has to do with transitions leaving a given state of the model only competing against the rest of transitions leaving that same state, instead of competing against all possible transitions in the model. CRFs provide the advantages of conditional models, and at the same time also solves the label bias problem.

CRFs fall under the category of discriminative classifiers, which, opposite to the generative models, represent the decision boundaries between the different classes considered. They are also modelled as undirected graphical models. In graphical models, a complex distribution over multiple variables is represented as the product of local functions that depend on smaller subsets of variables [221]. Undirected models, in particular, are families of probability distributions that each factorize based on a certain set of factors. The graph that will be used to model predictions depends on the task at hand. When the model has to predict a data point (one of the labels in the sequence), it also takes into account the previous label in the output sequence. This is done through the definition of what is called Feature Functions. These elements represent different characteristics extracted from the output label sequence (for example, in applications involving text processing, these could be words, or characters, or even the text layout). They are defined as:

$$f(X, i, y_{i-1}, y_i) \tag{A.8}$$

Where X is the set of inputs, i is the position in the dataset for which a prediction has to be obtained, y_{i-1} is the prediction obtained for the previous data point, and y_i is the label assigned to the current data point, Each feature function is based on the current observation and the label assigned to the previous observation. The conditional field is modelled as the normalized product of the feature function. This function has a set of weights that have to be learned by the model. In order to do this, they are estimated using the Maximum Likelihood Estimation (a method used to estimate the parameters in a probability distribution by maximizing a likelihood function that makes the observed data the most probable. Then, the parameters are optimized using Gradient Descent until the weights of the feature functions converge.

One of the fields of application where CRFs are more popular is the area of Natural Language Processing. This includes tasks like Part-of-Speech tagging (assigning to each word in a sentence its part of speech category: noun, verb, adverb, etc...) or Named Entity Recognition (identifying names referring to different entities, like persons, organizations, or locations). CRFs are used also in other tasks, like object recognition or image segmentation.

APPENDIX B

Detailed description of the software developed in this dissertation

This appendix details all the attributes and methods of the classes described in Figures 3.7 and 4.3.

B.1 Classes in the HRI Manager software architecture

B.1.1 CA Base thread class

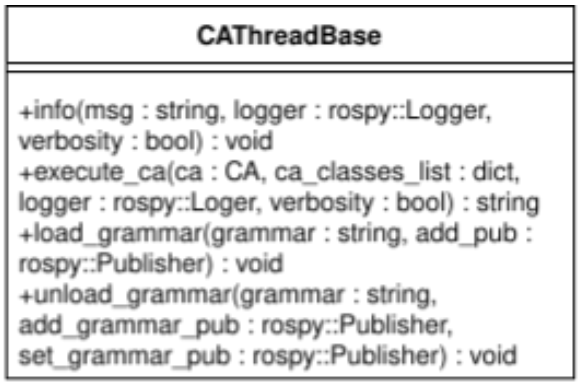


Figure B.1: Detailed view of the CAThreadBase class.

B.1.2 HRI Manager class

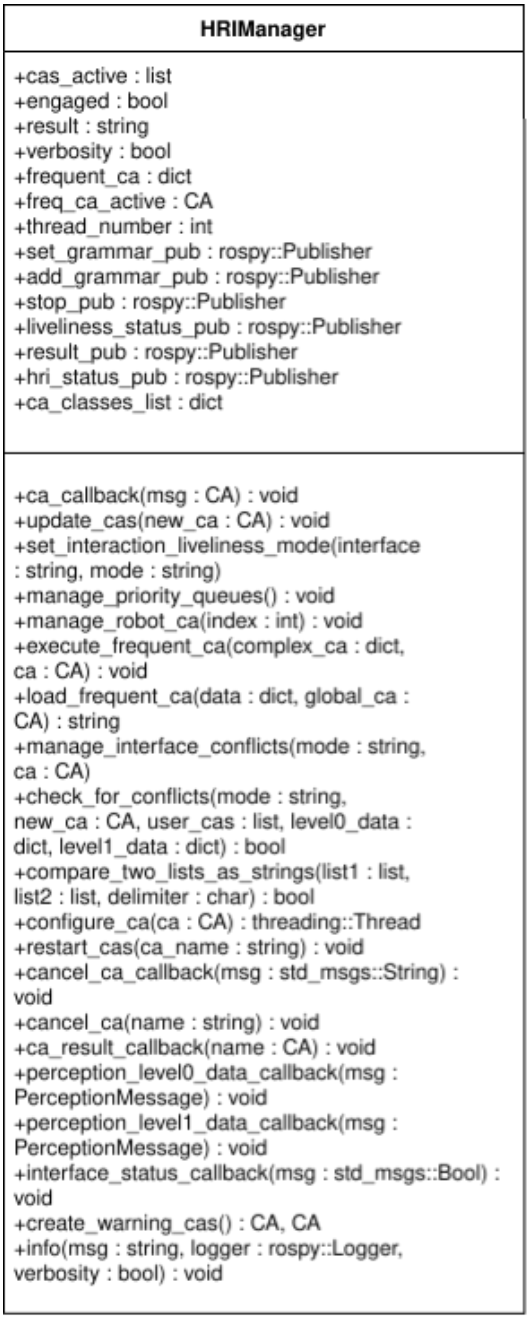


Figure B.2: Detailed view of the HRIManager class.

B.1.3 Immediate CA class

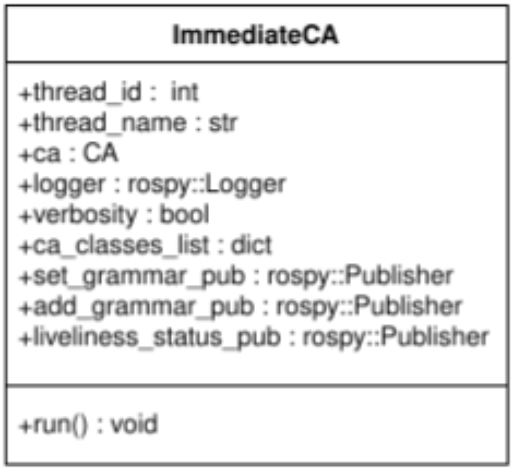


Figure B.3: Detailed view of the ImmediateCA class.

B.1.4 Continuous CA class

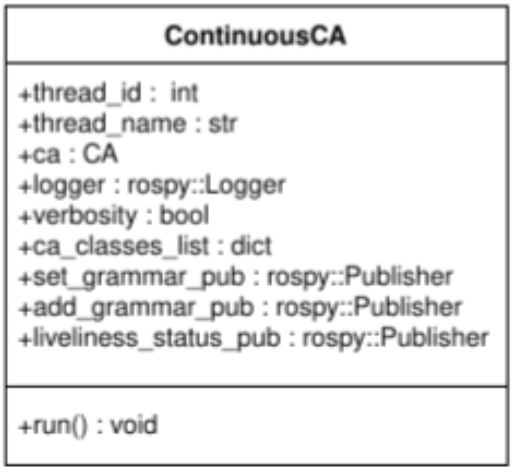


Figure B.4: Detailed view of the ContinuousCA class.

B.1.5 Communicative Act class

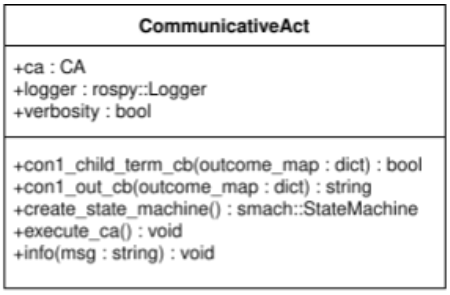


Figure B.5: Detailed view of the CommunicativeAct class.

B.1.6 Base State class

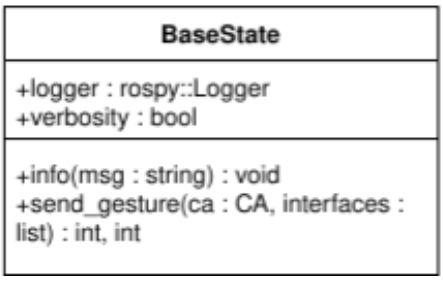


Figure B.6: Detailed view of the BaseState class.

B.1.7 State Send Data class

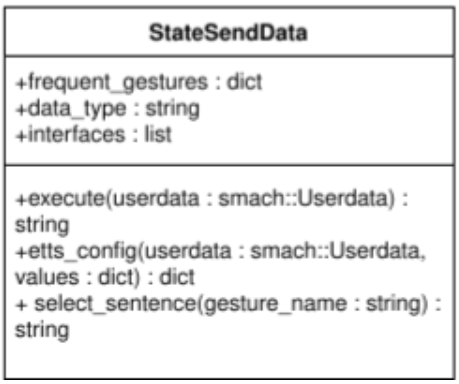


Figure B.7: Detailed view of the StateSendData class.

B.1.8 Robot Gives Information class

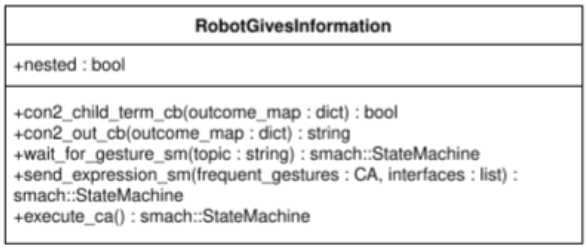


Figure B.8: Detailed view of the RobotGivesInformation class.

B.1.9 Robot Asks For Information class

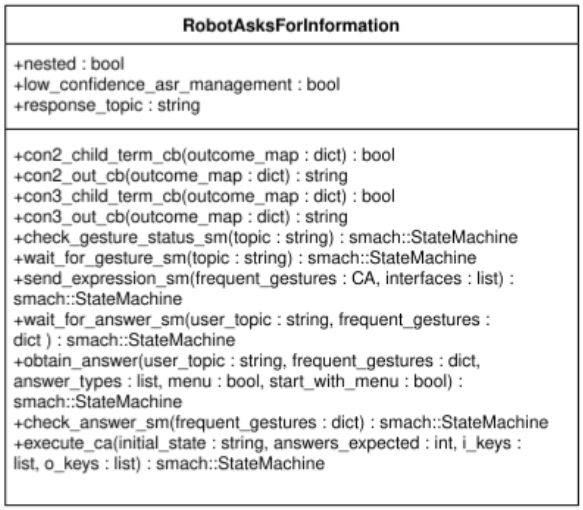


Figure B.9: Detailed view of the RobotAsksForInformation class.

B.1.10 User Gives Information class

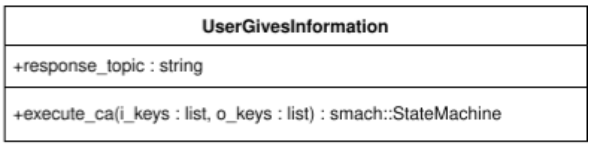


Figure B.10: Detailed view of the UserGivesInformation class.

B.1.11 User Asks For Information class

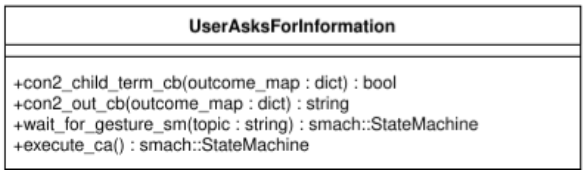


Figure B.11: Detailed view of the UserAsksForInformation class.

B.1.12 Question With Confirmation class

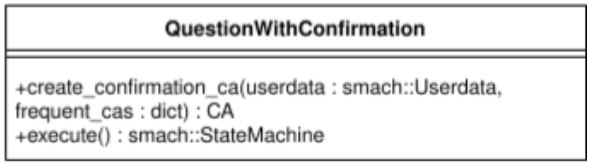


Figure B.12: Detailed view of the QuestionWithConfirmation class.

B.1.13 RightWrongQuestion class

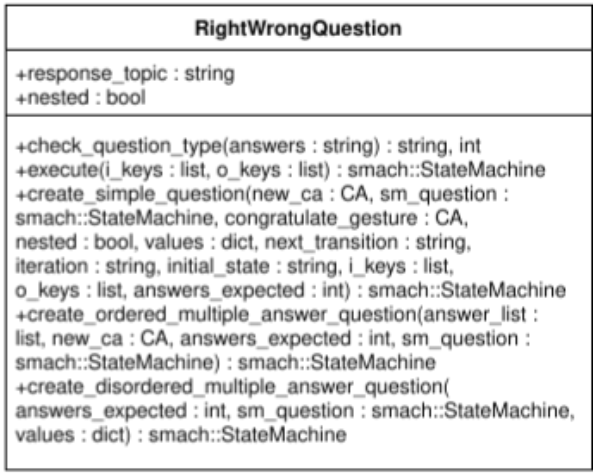


Figure B.13: Detailed view of the Right Wrong Question class.

B.1.14 Communication Warning class

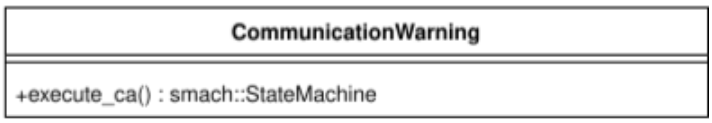


Figure B.14: Detailed view of the CommunicationWarning class.

B.1.15 Switching Mode Question class

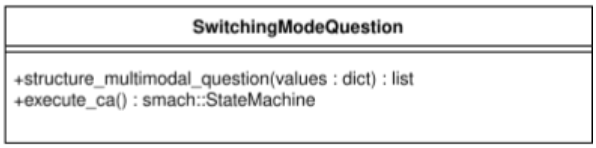


Figure B.15: Detailed view of the SwitchingModeQuestion class.

B.1.16 Manage Multimedia Content class

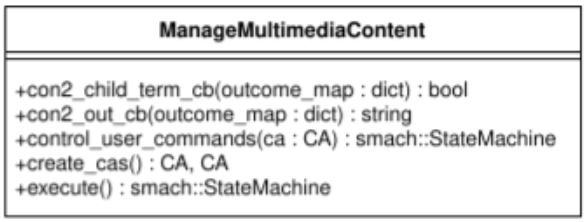


Figure B.16: Detailed view of the ManageMultimediaContent class.

B.2 Classes in the Expression Manager software architecture

B.2.1 Expression Scheduler class

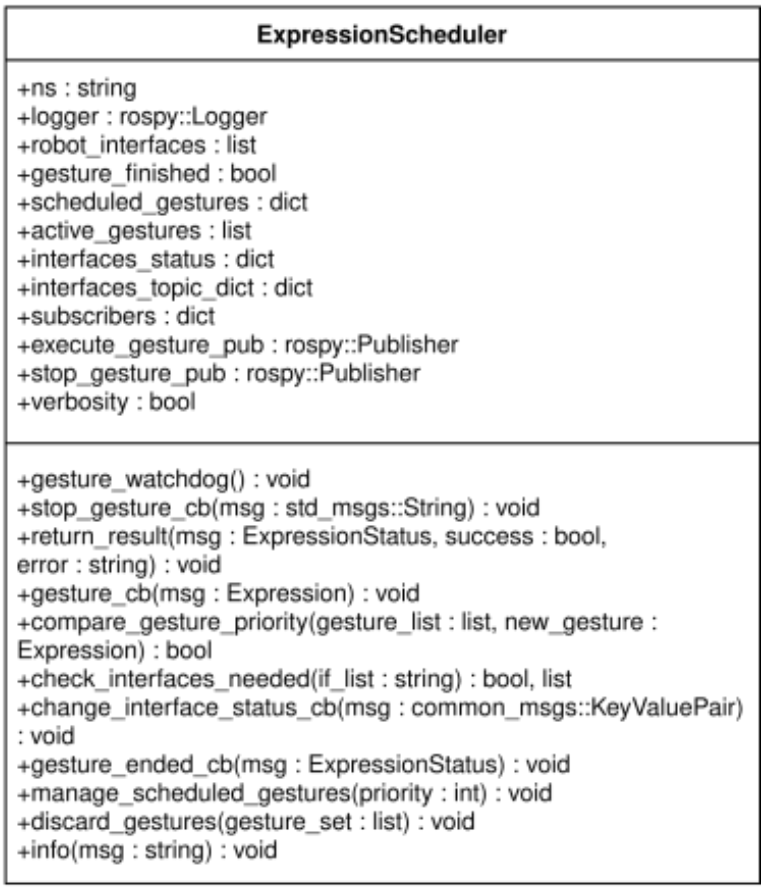


Figure B.17: Detailed view of the ExpressionScheduler class.

B.2.2 Expression Executor class

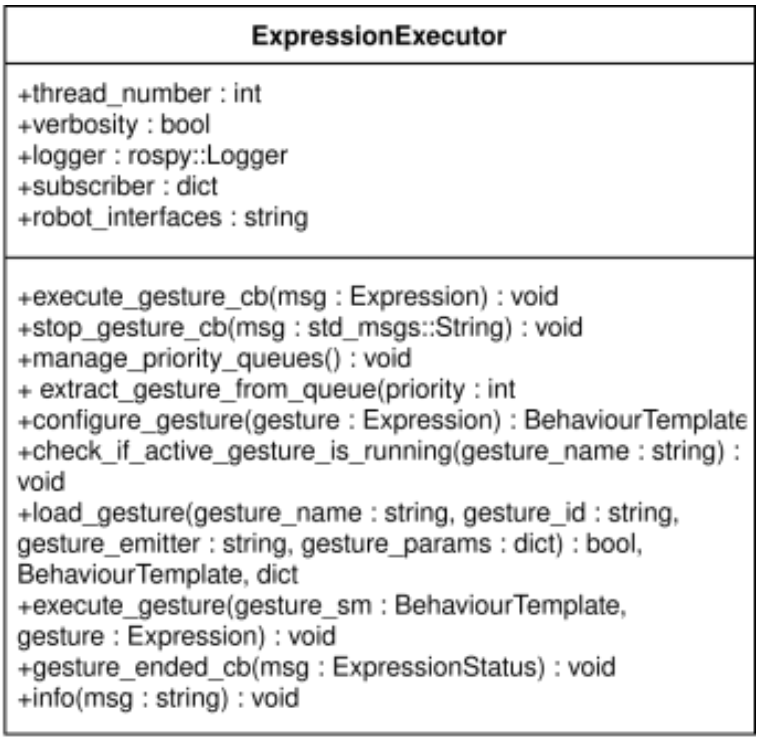


Figure B.18: Detailed view of the ExpressionExecutor class.

B.2.3 Gesture SM class

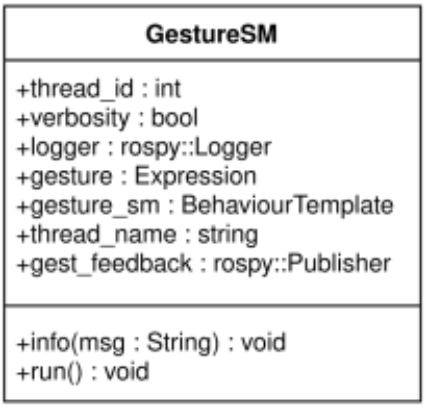


Figure B.19: Detailed view of the GestureSM class.

B.2.4 Behaviour Template class

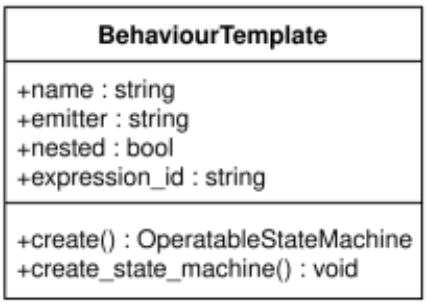


Figure B.20: Detailed view of the BehaviourTemplate class.

B.2.5 Speak class

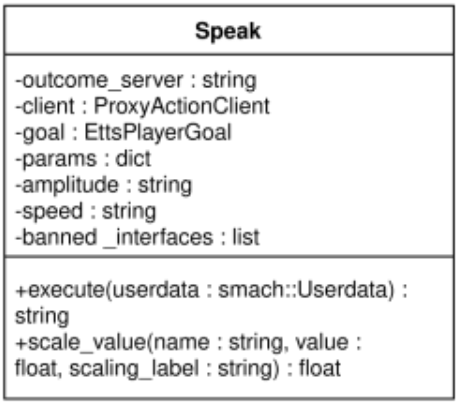


Figure B.21: Detailed view of the Speak class.

B.2.6 alz_angrySM class

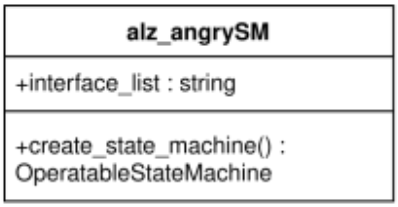


Figure B.22: Detailed view of the alz_angrySM class.

B.2.7 Joint Player class

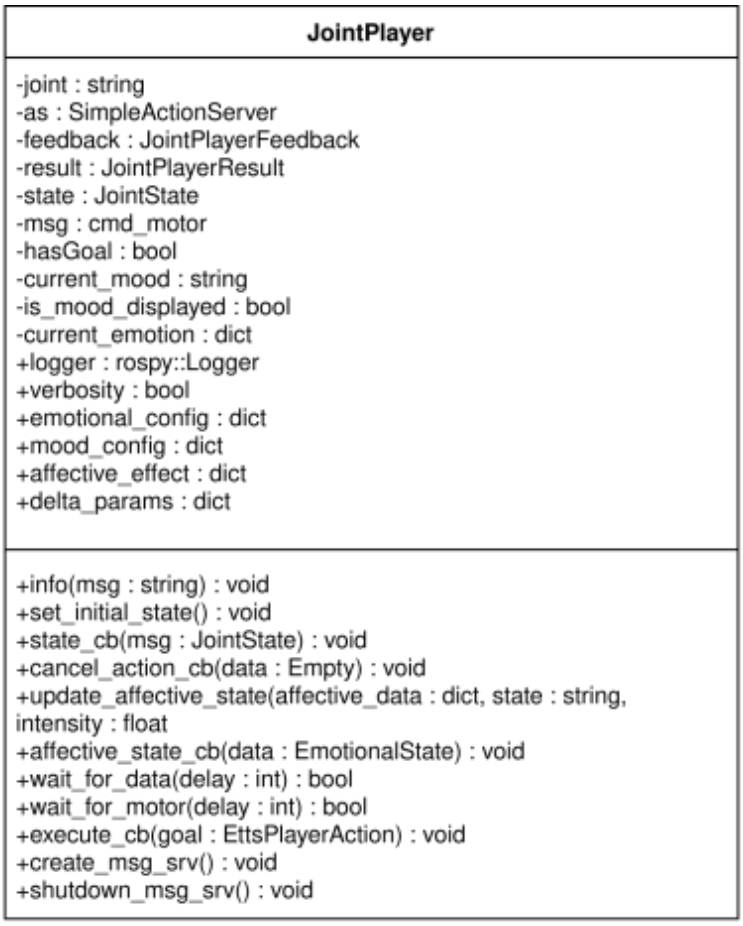


Figure B.23: Detailed view of the JointPlayer class.

B.2.8 Etts Player class

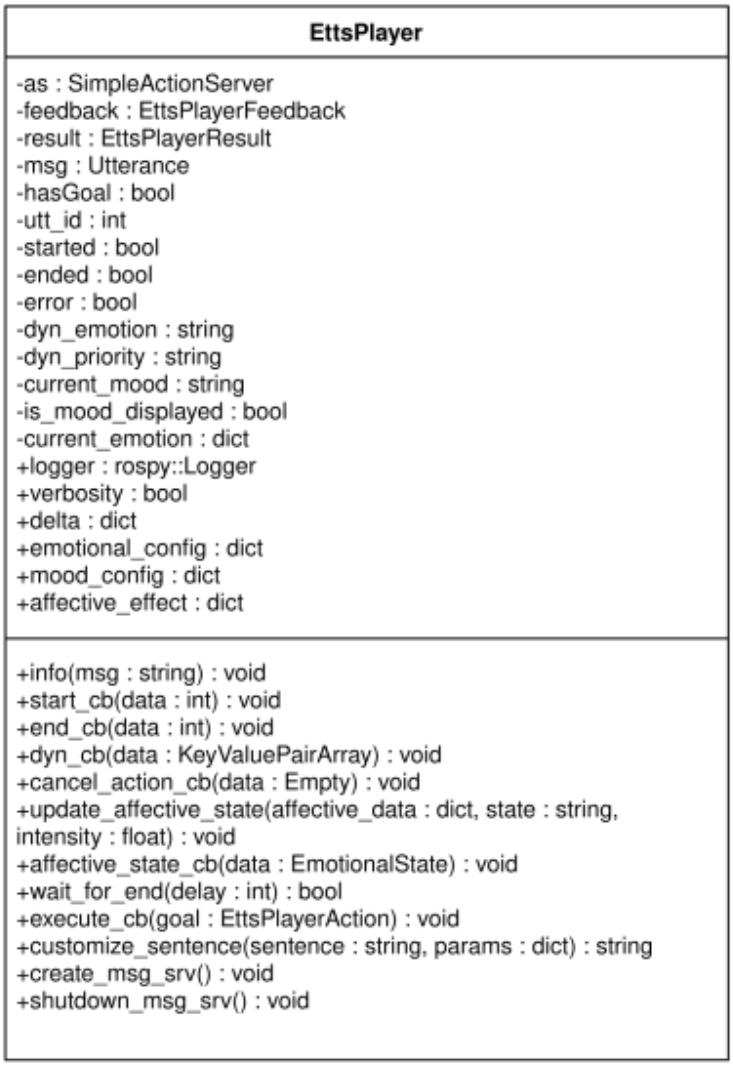


Figure B.24: Detailed view of the EttsPlayer class.

B.2.9 Touch Screen Player class

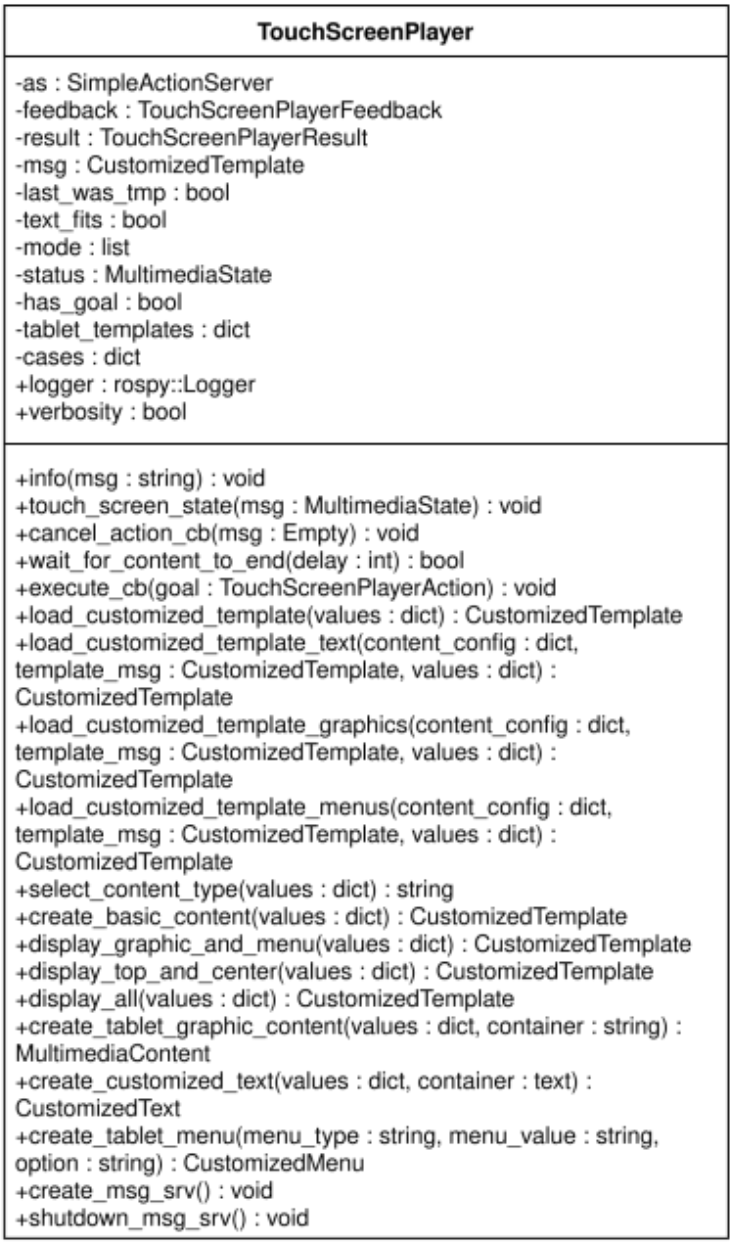


Figure B.25: Detailed view of the TouchScreenPlayer class.

B.2.10 LED Player class

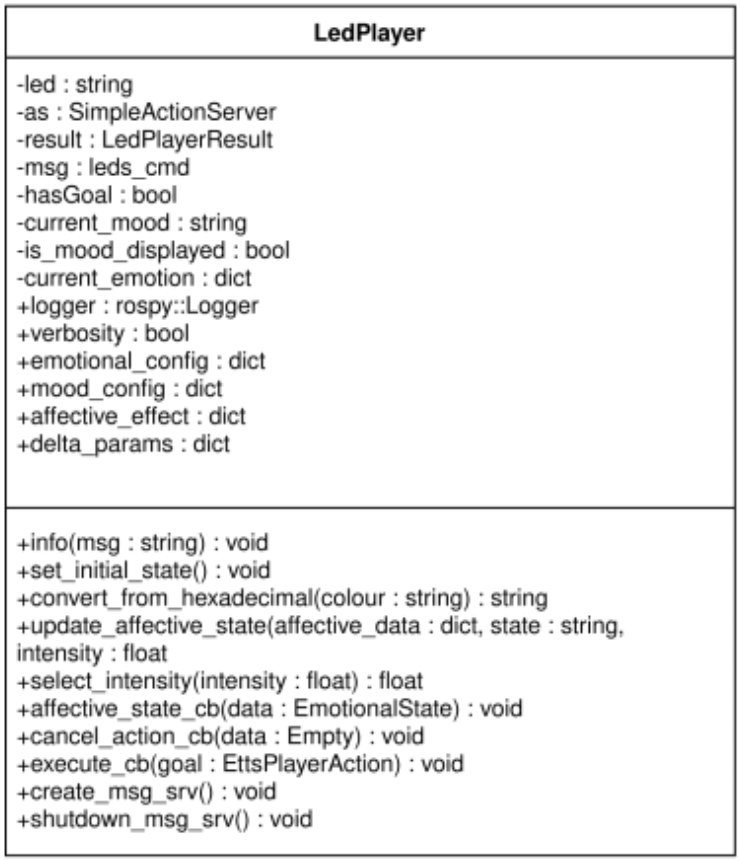


Figure B.26: Detailed view of the LedPlayer class.

B.2.11 Eye Player class

Class used to model the HRI Manager core

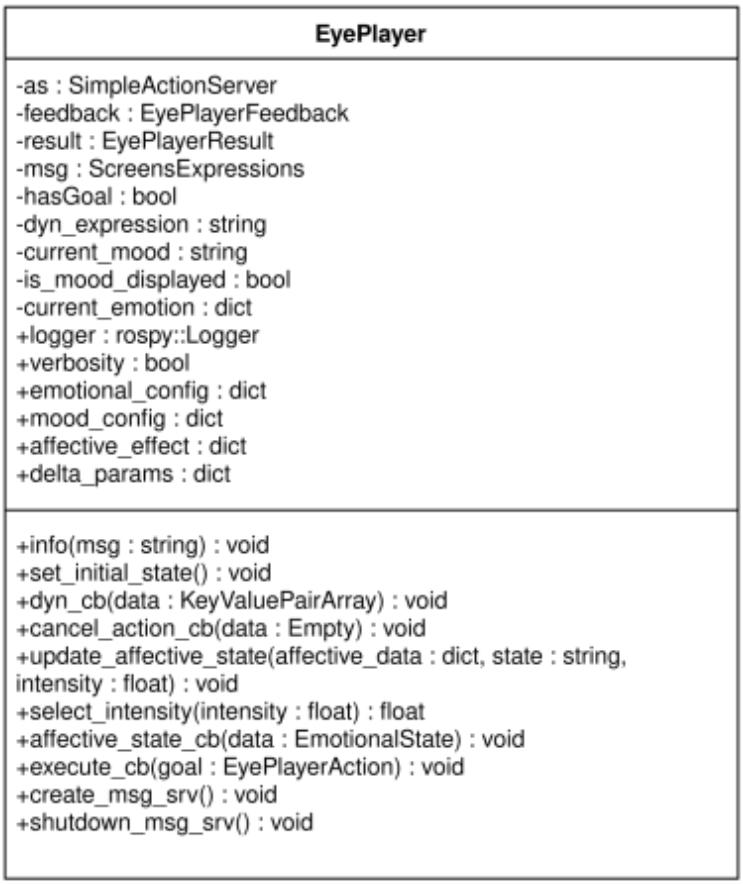


Figure B.27: Detailed view of the EyePlayer class.

APPENDIX C

Description of all the states developed for building multimodal expressions

This appendix describes in detail all the states that have been developed for building multimodal expressions in the expressiveness architecture described in Chapter 4. The inputs and outputs to each state (in the form of FlexBE userdata) are also presented. All the states share a common parameter:

- **behaviour_name (string):** The name of the expression where the states will be integrated. It will be used to create the communication channels used by the Expression Executor to control the execution of the gesture.

C.1 Action Control

This state controls when a specific Interface Player completes the goal that is currently being executed. The state checks the value of boolean variables that indicate if a Player is in use or not. If for a given Player, the value is True, that means that is currently executing an action. If it is False, that means that no action is being executed. The state waits for the value of the variable for the specified interface/s to change from True to False.

The following parameters can be configured:

- **interface (string):** interface of the robot that has to be monitored (eyes, etts, left arm...).
- **clear (bool):** parameter used for cleaning possible messages in the ROS topic. By default it should be set to False.

This state does not use specific userdata.

C.2 Color LED

This state is used to control the LED of the robot. Currently, it allows to control either the cheeks or the heart. The following parameters can be configured:

- **leds (string):** The name of the LED being controlled ('heart', 'cheeks').
- **color (string):** The colour of the LED, in hexadecimal value (0xRRGGBB).
- **min_intensity and max_intensity (int, optional):** Control the bright limits. Both are specified in percentage, this is, with a value between 0 and 100.
- **fade (bool, optional):** Boolean used to activate a fading effect.
- **fade_velocity (int, optional):** Controls the speed of the fading. Range: 0-100. Requires that the parameter fade is set to True.

The Color LED state uses the following userdata:

- **configuration_data (dict):** python dictionary that allows to modify the predefined values of the state's parameters. The keys in the dict are the name of the states that have to be modified, and the module is a serialized python dictionary (a dictionary converted to a string) containing the new configuration for the state. This parameter is shared between all states that require it, so it only needs to be defined once in the Expression.
- **banned_interfaces (string):** contains a string with the interfaces that cannot be used, separated by a delimiting character (Example: "etts|rightArm|cheeks").
- **amplitude (string):** used to scale the gesture's amplitude. In this case, this parameter modifies the maximum intensity. Seven possible values: high increase, medium increase, low increase, normal, low decrease, medium decrease, high decrease.

- **speed (string):** used to scale the gesture in time. In this case, this parameter modifies the fade_velocity. Seven possible values: high increase, medium increase, low increase, normal, low decrease, medium decrease, high decrease.
- **id (dictionary):** Identifier for the goal sent to the players that communicate with the robot's output interfaces. This userdata is for internal use only, and cannot be used by the developers.

C.3 Display Touch Screen

This state allows to display information through a touch screen (text, images, menus, or different combinations of all of them). The following parameters can be configured:

- **tablet_type (string):** Used to configure the type of the multimedia content that is going to be displayed ('image', 'gif').
- **tablet_url (string):** The path to the image or GIF being displayed. All images and GIFs have to be stored in the tablet.
- **menu_type (string):** Configures the type of content inside the buttons of a menu ('icon' for images, 'text' for text). If only one value is set, it applies to all the buttons. For configuring each button separately, there has to be one value per button, with the following format: "value1|value2|value3|...".
- **menu_value (string):** Value for each button. If the content being displayed in the buttons are images, the value has to be the path to the image. The format for the values is: "path_to_image1|text|text|path_to_image2|...".
- **tablet_text (string):** Text that is going to be displayed.
- **tablet_duration (float, required only if the content displayed is a video or an audio):** The amount of time the multimedia content has to be shown for.
- **tablet_init_time (float, optional):** Used to indicate if a multimedia content has to start at a specific point. If not specified, the content will be reproduced from the beginning. For example, it can be used to specify that a video has to start from the 2 minutes mark.
- **tablet_template (string, optional):** The robot's touch screen allows to use predefined templates to display content in the screen. This parameter is the name of the template that has to be used.

For multimedia content, only the values for tablet_type and tablet_url have to be set. For menus, only the values for menu_type and menu_value have to be set. The touch screen package allows also to display both multimedia content and a menu at the same time. In that case, the four parameters are required. The template can be selected automatically based on the number of buttons and, if menu_type is text, the number of characters per button. For more than one instance of each type of data (reproducing two videos or showing two images at the same time, for example), one of the predefined templates has to be specified.

The state uses the following userdata:

- **configuration_data (dict):** python dictionary that allows to modify the predefined values of the state's parameters. The keys in the dict are the name of the states that have to be modified, and the module is a serialized python dictionary (a dictionary converted to a string) containing the new configuration for the state. This parameter is shared between all states that require it, so it only needs to be defined once in the Expression.
- **banned_interfaces (string):** contains a string with the interfaces that cannot be used, separated by a delimiting character (Example: "etts|rightArm|cheeks").
- **id (dictionary):** Identifier for the goal sent to the players that communicate with the robot's output interfaces. This userdata is for internal use only, and cannot be used by the developers.

C.4 Express Eyes

This state allows to send commands to the screens that serve as eyes for our robots. The following parameters can be configured:

- **expression (int):** The expression that the eyes have to convey (sadness, happiness, suspicion, drowsiness...).
- **orientation (int, optional):** Modifies in which direction the robot is looking. There are nine possible directions (eight of them placed in $45\ \mathrm{deg}$ intervals and a ninth one where the gaze is kept forward)
- blink_frequency (int, optional): Modifies the frequency used for blinking (the period between blinks in milliseconds)
- pupil_shape (int, optional): Changes the shape of the pupil. A predefined set of shapes can be used. This functionality is only implemented in Gero, and is not available on Mini

- pupil_size (int, optional): Changes the size of the pupil. It can take values from 0 to 100. This functionality is only implemented in Gero, and is not available on Mini
- blink_speed (int, optional): Changes the blinking speed (how long does it take to close and open the eyes once). It can take values from 1 to 95 (if the value is set to 0, the current blink speed is kept). This functionality is only implemented in Gero, and is not available on Mini.
- eyes_desviation (int, optional): the desviation of the eyes, which can produce from strabismus to crossed eyes. It can take values from 0 to 100. This functionality is only implemented in Gero, and is not available on Mini.
- lid_colour, sclera_colour, and iris_colour (int, optional): changes the colour of the lids, the sclera and the iris. A predefined pallet of colours can be used. This functionality is only implemented in Gero, and is not available on Mini.

The state uses the following userdata:

- **configuration_data (dict):** python dictionary that allows to modify the predefined values of the state's parameters. The keys in the dict are the name of the states that have to be modified, and the module is a serialized python dictionary (a dictionary converted to a string) containing the new configuration for the state. This parameter is shared between all states that require it, so it only needs to be defined once in the Expression.
- **banned_interfaces (string):** contains a string with the interfaces that cannot be used, separated by a delimiting character (Example: "etts|rightArm|cheeks").
- **speed (string):** used to scale the gesture in time. In this case, this parameter modifies the blinking frequency. Seven possible values: high increase, medium increase, low increase, normal, low decrease, medium decrease, high decrease.
- **id (dictionary):** Identifier for the goal sent to the players that communicate with the robot's output interfaces. This userdata is for internal use only, and cannot be used by the developers.

C.5 For Loop

This state allows to implement a *for* loop inside a gesture. It receives the number of iterations and checks if the iteration limit has been reached every time the state machine executes this state. Every time the state is executed, the number of iterations is increased by one. The following parameters can be configured:

- **iterations (int):** The number of iterations that the loop has to perform.

The state uses the following userdata:

- **iterations (int):** contains the number of iterations already performed. This userdata is only for internal use only, and cannot be used by the developers. It has to be set to 0 when defining the userdata used by the gesture.

C.6 If Loop

Allows to implement an *if* loop in a gesture. This state checks if a condition is met for a specific parameter and selects a different outcome depending on the result of this operation. Currently, it only allows two different outcomes: either the condition is met, or it is not. The following parameters can be configured:

- **param (string):** The name of the parameter whose value the state is going to check in order to see if the condition is met. This parameter has to be stored in the ROS parameter server.
- **condition (undefined type):** The condition that has to be validated. If the value stored in *param* matches the value defined for this parameter, then the condition is met.

This state does not use any userdata.

C.7 Move Joint

This state allows to control the robot's motors. The following parameters can be configured:

- **joint_name (string):** Name of the motor receiving the command (for Mini: *leftArm*, *rightArm*, *head*, *neck*, *base*. Gero shares the same motors, except for the *base* motor).
- **value (float):** position for the motor (in radians). A full trajectory can be specified as a string like this: "0.2|0.7|0.3|1.5|0.0". The player will send each of the positions in order, once the previous one has been reached.
- **relative_mvt (bool, optional):** If true, the motor will add the position defined in the *value* parameter to the current position of the motor in order to find the final position (if the motor is in 1 and the value is 0.5, the motor will move to the position 1.5). If false (default value),

the position stored in *value* will be treated as a absolute position (in the previous example, the motor would move to the position 0.5).

- **velocity (float, optional):** Configures the speed of the motor (Not all robots have speed control implemented).
- **acceleration (float, optional):** Configures the acceleration of the motor (Not all robots have acceleration control implemented).

The state uses the following userdata:

- **configuration_data (dict):** python dictionary that allows to modify the predefined values of the state's parameters. The keys in the dict are the name of the states that have to be modified, and the module is a serialized python dictionary (a dictionary converted to a string) containing the new configuration for the state. This parameter is shared between all states that require it, so it only needs to be defined once in the Expression.
- **banned_interfaces (string):** contains a string with the interfaces that cannot be used, separated by a delimiting character (Example: "etts|rightArm|cheeks").
- **amplitude (string):** used to scale the gesture's amplitude. In this case, this parameter modifies the final position. Seven possible values: high increase, medium increase, low increase, normal, low decrease, medium decrease, high decrease.
- **speed (string):** used to scale the gesture in time. In this case, this parameter modifies the speed and acceleration. Seven possible values: high increase, medium increase, low increase, normal, low decrease, medium decrease, high decrease.
- **id (dictionary):** Identifier for the goal sent to the players that communicate with the robot's output interfaces. This userdata is for internal use only, and cannot be used by the developers.

C.8 Reset Interfaces

This state is used to reset all the robot's interfaces to their default state (for example, the head has to be looking forward, and the arms have to be at the sides of the body). This state is used in the template from which all the expressions inherit. It receives one parameter:

- **interfaces_list (string):** contains a string with the interfaces that have to be reset, separated by a delimiting character (Example: "etts|rightArm|cheeks").

This state does not use any userdata.

C.9 Return Result

This state is used to return the result of the gesture's execution to the rest of the modules of the robot's software architecture. This state is not used by the developers of expressions, and is instead integrated in the template from which all the expressions inherit. It receives the following parameters:

- **expression_id (string):** the ID of the expression where the state is used.
- **result (string):** string that indicates the result of the expression's execution.

This state does not use any userdata.

C.10 Select Random Gesture

This state receives a list of possible gestures and selects one randomly. It receives one parameter:

- **gestures (string):** string containing all the gestures among which the state has to choose one. The format is the following: "gesture_1|gesture_2|gesture_3|..."

The state uses the following userdata:

- **next_gesture (string):** contains the name of the gesture selected by the state. This userdata is for internal use only, and cannot be used by the developers.

C.11 Execute Random Gesture

This state loads and executes a gesture received through userdata. In order to use this state, the gesture needs to contain also at least one instance of the state *Select Random Gesture*, which has to be executed before this one (it could be replaced by an equivalent state that picks a gesture and stores it in the userdata next_gesture). It does not receive any parameter (besides those common to all states). The state uses the following userdata:

- **next_gesture (string):** contains the name of the gesture that is going to be executed. This userdata is for internal use only, and cannot be used by the developers.

C.12 Skill Start

This state starts one of the skills of the robot (for example, a face tracking skill) and sends a goal to that skill. In order to be used in expressions, the skill has to follow a particular template. The following parameters can be configured:

- **skill_name (string):** Contains the name of the skill that has to be activated. The name has to have the same format used in the communication channels of the skill.
- **msg_type (string):** The name of the ActionLib message used to send goals to the skill. It is usually located in the action folder inside the ROS package containing the skill.
- **values (dictionary):** A python dictionary containing all the required information to configure the goal sent to the skill.

This state does not use any userdata.

C.13 Skill Stop

Deactivates an active skill. In order to be used in expressions, the skill has to follow the skill template (see documentation). The following parameters can be configured:

- **skill_name (string):** Contains the name of the skill that has to be deactivated. The name has to have the same format used in the communication channels of the skill.

This state does not use any userdata.

C.14 Speak

This state sends commands to the TTS player in order to request that the robot utters one or more sentences. The following parameters can be configured:

- **text (string):** The sentence the robot has to say. There are specific items that can be used to modulate the sentence, depending on the TTS used (for example, the *pause item can be used to add a pause to the utterance*).

- **language (string, optional):** Changes the language of the TTS. It should match the language used in the sentence, but this is not required.
- **pitch (int, optional):** Raises or lowers the pitch. The value represents a percentage increase or decrease over the current pitch. It can take values from -5 to 10. Not all robots allow to modulate the pitch.
- **speed (float, optional):** Changes the rate of articulation (the speed of the voice). It can take values from 0.65 to 0.85. Not all robots allow rate of articulation modification.
- **primitive (int, optional):** Selects the TTS that will be used. Each one offers different features.
- **emotion (int, optional):** Modifies the emotion used for the sentence. The emotion can be selected from a limited set of options.
- **volume (int, optional):** Modifies the volume of the voice. It can take values from 0 to 100.
- **priority (string, optional):** The priority for the sentence. It defines what the TTS does if it receives a new utterance while there is another one being said (for example, queue the new one or cancel the current utterance and process the next one).

The state uses the following userdata:

- **configuration_data (dict):** python dictionary that allows to modify the predefined values of the state's parameters. The keys in the dict are the name of the states that have to be modified, and the module is a serialized python dictionary (a dictionary converted to a string) containing the new configuration for the state. This parameter is shared between all states that require it, so it only needs to be defined once in the Expression.
- **banned_interfaces (string):** contains a string with the interfaces that cannot be used, separated by a delimiting character (Example: "etts|rightArm|cheeks").
- **amplitude (string):** used to scale the gesture's amplitude. In this case, this parameter modifies the volume of the voice. Seven possible values: high increase, medium increase, low increase, normal, low decrease, medium decrease, high decrease.
- **id (dictionary):** Identifier for the goal sent to the players that communicate with the robot's output interfaces. This userdata is for internal use only, and cannot be used by the developers.

Bibliography

[1] "How robots change the world," tech. rep., Oxford Economics, Oxford, UK, June 2019.

[2] Euronews, "Meet little casper, a robot designed to help children suffering from cancer." `https://www.euronews.com/2016/03/28/meet-little-casper-a-robot-designed-to-help-children-suffering-from-cancer`. Accessed: 2021-01-27.

[3] M. R. Keyvanpour, M. B. Shirzad, and H. Rashidghalam, "Afdm: an analytical framework of dialogue manager," in *2020 8th Iranian Joint Congress on Fuzzy and intelligent Systems (CFIS)*, pp. 148–153, IEEE, 2020.

[4] A. Dobra, "General classification of robots. size criteria," in *2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, pp. 1–6, 2014.

[5] P. Marešová, H. Mohelská, and K. Kuča, "Economics aspects of ageing population," *Procedia Economics and Finance*, vol. 23, pp. 534 – 538, 2015. 2nd GLOBAL CONFERENCE on BUSINESS, ECONOMICS, MANAGEMENT and TOURISM.

[6] A. Tinker, "The social implications of an ageing population," *Mechanisms of Ageing and Development*, vol. 123, no. 7, pp. 729 – 735, 2002. The Biology of Ageing.

[7] T. Dall, P. Gallo, R. Chakrabarti, T. West, A. Semilla, and M. Storm, "An aging population and growing disease burden will require alarge and specialized health care workforce by 2025," *Health affairs (Project Hope)*, vol. 32, pp. 2013–20, 11 2013.

[8] C. Breazeal, "Toward sociable robots," *Robotics and Autonomous Systems*, vol. 42, no. 3, pp. 167 – 175, 2003. Socially Interactive Robots.

[9] C. Bartneck and J. Forlizzi, "A design-centred framework for social human-robot interaction," pp. 591 – 594, 10 2004.

[10] S. Hutson, S. L. Lim, P. Bentley, N. Bianchi-Berthouze, and A. Bowling, "Lecture notes in computer science," pp. 578–587, 01 2011.

[11] K. Winkle, P. Caleb-Solly, A. Turton, and P. Bremner, "Social robots for engagement in rehabilitative therapies: Design implications from a study with therapists," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '18, (New York, NY, USA), p. 289–297, Association for Computing Machinery, 2018.

[12] E. J. Rose and E. A. Björling, "Designing for engagement: Using participatory design to develop a social robot to measure teen stress," in *Proceedings of the 35th ACM International Conference on the Design of Communication*, SIGDOC '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[13] P. Lison, *Structured Probabilistic Modelling for Dialogue Management*. PhD thesis, Department of Informatics, Faculty of Mathematics and Natural Sciences, University of Oslo, Oslo, Norway, Oct. 2013.

[14] M. Hoy, "Alexa, siri, cortana, and more: An introduction to voice assistants," *Medical Reference Services Quarterly*, vol. 37, pp. 81–88, 01 2018.

[15] N. Roy, J. Pineau, and S. Thrun, "Spoken dialogue management using probabilistic reasoning," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, (USA), p. 93–100, Association for Computational Linguistics, 2000.

[16] J. Gustafson, P. Elmberg, R. Carlson, and A. Jönsson, "An educational dialogue system with a user controllable dialogue manager," in *In Proceedings of ICSLP'98*, pp. 33–37, 1998.

[17] G. Di Fabrizzio and C. Lewis, "Florence: a dialogue manager framework for spoken dialogue systems," in *ICSLP 2004, 8th International Conference on Spoken Language Processing , Jeju, Jeju Island, Korea*, pp. 4–8.

[18] R. Riggio and H. Friedman, "Impression formation. the role of expressive behavior," *Journal of personality and social psychology*, vol. 50, pp. 421–7, 03 1986.

[19] E. Fromm, *The Heart of Man: Its Genius for Good and Evil*. New York, NY: Harper & Row, 1964.

[20] E. O. Wilson, *Biophilia*. Cambridge, MA: Harvard University Press, 1984.

[21] T. Wheatley and A. Martín, *The neuroscience of social interaction*, pp. 345–353. Lippincott, Williams & Wilkins, 2009.

[22] G. Melson, A. Beck, and B. Friedman, "Robotic pets in human lives: Implications for the human-animal bond and for human relationships with personified technologies," *Journal of Social Issues - J SOC ISSUES*, vol. 65, pp. 545–567, 09 2009.

[23] C. Bartneck, T. Kanda, O. Mubin, and A. Mahmud, "Does the design of a robot influence its animacy and perceived intelligence?," *International Journal of Social Robotics*, vol. 1, pp. 195–204, 04 2009.

[24] M. Nakayama and S. Yamanaka, "Perception of animacy by the linear motion of a group of robots," in *Proceedings of the Fourth International Conference on Human Agent Interaction*, HAI '16, (New York, NY, USA), p. 3–9, Association for Computing Machinery, 2016.

[25] A. Castro-González, H. Admoni, and B. Scassellati, "Effects of form and motion on judgments of social robots' animacy, likability, trustworthiness and unpleasantness," *International Journal of Human-Computer Studies*, vol. 90, pp. 27 – 38, 2016.

[26] M. A. Salichs, R. Barber, A. M. Khamis, M. Malfaz, J. F. Gorostiza, R. Pacheco, R. Rivas, A. Corrales, E. Delgado, and D. Garcia, "Maggie: A robotic platform for human-robot social interaction," in *2006 IEEE Conference on Robotics, Automation and Mechatronics*, pp. 1–7, 2006.

[27] V. Gonzalez Pacheco, A. Castro-González, M. Malfaz, and M. A. Salichs, "Human-robot interaction in the monarch project," tech. rep., Robocity 2030 13th Workshop EU Robotic Projects Results, Madrid, Spain, Dec. 2015.

[28] M. A. Salichs, A. Castro-González, E. Salichs, and et. al., "Mini: A new social robot for the elderly," *International Journal of Social Robotics*, vol. 12, pp. 1231–1249, 2020.

[29] M. Maroto-Gómez, A. Castro-González, J. C. Castillo, M. Malfaz, and M. A. Salichs, "A bio-inspired motivational decision making system for social robots based on the perception of the user," *Sensors*, vol. 18, no. 8, pp. 2691–2710, 2018.

[30] S. González-Díaz, E. Velázquez Navarro, F. Alonso-Martín, A. Castro-González, J. Castillo, M. Malfaz, and M. Salichs, "El robot social mini como plataforma de información y ocio," 06 2019.

[31] E. Velázquez Navarro, S. González-Díaz, F. Alonso-Martín, J. Castillo, A. Castro-González, M. Malfaz, and M. Salichs, "El robot social mini como plataforma para el desarrollo de juegos de interacción multimodales," 06 2019.

[32] J. J. Gamboa-Montero, F. Alonso-Martín, J. C. Castillo, M. Malfaz, and M. A. Salichs, "Detecting, locating and recognising human touches in social robots with contact microphones," *Engineering Applications of Artificial Intelligence*, vol. 92, p. 103670, 2020.

[33] T. Elwert, "Continuous and explicit dialogue modelling," in *Conference Companion on Human Factors in Computing Systems*, pp. 265–266, 1996.

[34] M. Green, "A survey of three dialogue models," *ACM Transactions on Graphics (TOG)*, vol. 5, no. 3, pp. 244–275, 1986.

[35] R. E. Banchs and H. Li, "Iris: a chat-oriented dialogue system based on the vector space model," in *Proceedings of the ACL 2012 System Demonstrations*, pp. 37–42, 2012.

[36] T.-H. Wen, D. Vandyke, N. Mrksic, M. Gasic, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, and S. Young, "A network-based end-to-end trainable task-oriented dialogue system," *arXiv preprint arXiv:1604.04562*, 2016.

[37] J. Pei, P. Ren, and M. de Rijke, "A modular task-oriented dialogue system using a neural mixture-of-experts," *arXiv preprint arXiv:1907.05346*, 2019.

[38] K. Liao, Q. Liu, Z. Wei, B. Peng, Q. Chen, W. Sun, and X. Huang, "Task-oriented dialogue system for automatic disease diagnosis via hierarchical reinforcement learning," *arXiv preprint arXiv:2004.14254*, 2020.

[39] J. Bang, H. Noh, Y. Kim, and G. G. Lee, "Example-based chat-oriented dialogue system with personalized long-term memory," in *2015 International Conference on Big Data and Smart Computing (BIGCOMP)*, pp. 238–243, IEEE, 2015.

[40] H. Mori and M. Araki, "Selection method of an appropriate response in chat-oriented dialogue systems," in *Proceedings of the 17th annual meeting of the special interest group on discourse and dialogue*, pp. 228–231, 2016.

[41] A. I. Niculescu and R. E. Banchs, "Strategies to cope with errors in human-machine spoken interactions: using chatbots as back-off mechanism for task-oriented dialogues," *Proceedings of ERRARE, Sinaia, Romania*, 2015.

[42] B. R. Ranoliya, N. Raghuwanshi, and S. Singh, "Chatbot for university related faqs," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1525–1530, IEEE, 2017.

[43] M. Dahiya, "A tool of conversation: Chatbot," *International Journal of Computer Sciences and Engineering*, vol. 5, no. 5, pp. 158–161, 2017.

[44] D. Spiliotopoulos, I. Androutsopoulos, and C. D. Spyropoulos, "Human-robot interaction based on spoken natural language dialogue," in *Proceedings of the European workshop on service and humanoid robots*, pp. 25–27, 2001.

[45] P. Fodor, "Dialog management for decision processes," in *Proc. of the 3rd Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, 2007.

[46] D. R. Traum and S. Larsson, "The information state approach to dialogue management," in *Current and new directions in discourse and dialogue*, pp. 325–353, Springer, 2003.

[47] B. H. Trung, "Multimodal dialogue management-state of the art," *CTIT Technical Report Series*, no. 06-01, 2006.

[48] J. Peltason and B. Wrede, "Pamini: A framework for assembling mixed-initiative human-robot interaction from generic interaction patterns," in *Proceedings of the SIGDIAL 2010 Conference*, pp. 229–232, 2010.

[49] J. Peltason, *Modeling Human-Robot Interaction Interaction Based on Generic Interaction Patterns*. PhD thesis, Bielefeld University, Bielefeld, Germany, Dec. 2013.

[50] B. Souvignier, A. Kellner, B. Rueber, H. Schramm, and F. Seide, "The thoughtful elephant: Strategies for spoken dialog systems," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 51–62, 2000.

[51] F. Alonso-Martín, A. Castro-González, F. J. F. d. G. Luengo, and M. Á. Salichs, "Augmented robotics dialog system for enhancing human–robot interaction," *Sensors*, vol. 15, no. 7, pp. 15799–15829, 2015.

[52] C. Rich and C. L. Sidner, "Collagen: A collaboration manager for software interface agents," in *Computational models of mixed-initiative interaction*, pp. 149–184, Springer, 1998.

[53] M. Wessel, G. Acharya, J. Carpenter, and M. Yin, "An ontology-based dialogue management system for virtual personal assistants," in *International Workshop on Spoken Dialogue Systems Technology, Farmington, PA*, p. 12, June.

[54] M. Wahde, "A dialogue manager for task-oriented agents based on dialogue building-blocks and generic cognitive processing," in *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, pp. 1–8, IEEE, 2019.

[55] F. Morbini, D. DeVault, K. Sagae, J. Gerten, A. Nazarian, and D. Traum, "Flores: A forward looking, reward seeking, dialogue manager," in *4th International Workshop on Spoken Dialogue Systems, Ermenonville, France*, pp. 151–162, Nov.

[56] P. Lison, "A hybrid approach to dialogue management based on probabilistic rules," *Computer Speech & Language*, vol. 34, no. 1, pp. 232–255, 2015.

[57] P. Milhorat, D. Lala, K. Inoue, T. Zhao, M. Ishida, K. Takanashi, S. Nakamura, and T. Kawahara, "A conversational dialogue manager for the humanoid robot erica," in *Advanced Social Interaction with Agents*, pp. 119–131, Springer, 2019.

[58] B. Kiefer, A. Welker, and C. Biwer, "Vonda: A framework for ontology-based dialogue management," *arXiv preprint arXiv:1910.00340*, 2019.

[59] J. L. Austin, *How to do things with words*, vol. 88. Oxford university press, 1975.

[60] J. Chu-Carroll, "Mimic: An adaptive mixed initiative spoken dialogue system for information queries," in *Sixth Applied Natural Language Processing Conference*, pp. 97–104, 2000.

[61] D. Bohus and A. I. Rudnicky, "The ravenclaw dialog management framework: Architecture and systems," *Computer Speech & Language*, vol. 23, no. 3, pp. 332–361, 2009.

[62] M. Turunen and J. Hakulinen, "Agent-based adaptive interaction and dialogue management architecture for speech applications," in *International Conference on Text, Speech and Dialogue*, pp. 357–364, Springer, 2001.

[63] T. Nestorovič, "Creating a general collaborative dialogue agent with lounge strategy feature," *Expert Systems with Applications*, vol. 39, no. 2, pp. 1607–1625, 2012.

[64] P.-H. Su, M. Gasic, N. Mrksic, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young, "Continuously learning neural dialogue management," *arXiv preprint arXiv:1606.02689*, 2016.

[65] H. Cuayáhuitl, S. Yu, A. Williamson, and J. Carse, "Deep reinforcement learning for multi-domain dialogue systems," *arXiv preprint arXiv:1611.08675*, 2016.

[66] T.-H. Wen, D. Vandyke, N. Mrksic, M. Gasic, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, and S. Young, "A network-based end-to-end trainable task-oriented dialogue system," in *15th Conference of the European Chapter of the Association for Computational Linguistics*, vol. 1, pp. 438–449, Apr. 2017.

[67] X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz, "End-to-end task-completion neural dialogue systems," in *8th International Conference on Natural Language Processing*, pp. 733–743, Nov. 2017.

[68] M. Eric and C. D. Manning, "Key-value retrieval networks for task-oriented dialogue," in *SIGDIAL 2017 Conference*, pp. 37–49, Aug. 2017.

[69] L. Xu, Q. Zhou, K. Gong, X. Liang, J. Tang, and L. Lin, "End-to-end knowledge-routed relational dialogue system for automatic diagnosis," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 7346–7353, 2019.

[70] H. Xu, H. Peng, H. Xie, E. Cambria, L. Zhou, and W. Zheng, "End-to-end latent-variable task-oriented dialogue system with exact log-likelihood optimization," *World Wide Web*, pp. 1–14, 2019.

[71] T. Zhao, "Reinforest: Multi-domain dialogue management using hierarchical policies and knowledge ontology," tech. rep., CMU-LTI-16-006. Language Technologies Institute Carnegie Mellon University, 2016.

[72] T.-H. Lin, T. Bui, D. S. Kim, and J. Oh, "A multimodal dialogue system for conversational image editing," in *32nd Conference on Neural Information Processing Systems*, Aug. 2018.

[73] T. Saha, S. Saha, and P. Bhattacharyya, "Towards sentiment aided dialogue policy learning for multi-intent conversations using hierarchical reinforcement learning," *PloS one*, vol. 15, no. 7, p. e0235367, 2020.

[74] C. Cherry, "On human communication," 1966.

[75] J. L. Austin, *How to do things with words*, vol. 88. Oxford university press, 1962.

[76] Blackburn, Simon W., "Philosophy of language. encyclopedia britannica." `https://www.britannica.com/topic/philosophy-of-language`. Accessed: 2021-01-27.

[77] W. G. Lycan, *Philosophy of Language: a Contemporary Introduction*. Taylor & Francis Group, New York, NY, 2000.

[78] L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, and J. B. Reece, *Philosophy of language*, pp. 8–26. New York, NY: Routledge, 1997.

[79] R. D. Van Valin, *An Introduction to Syntax*. Cambridge university press Cambridge, UK, 2001.

[80] G. Leech, *Principles of pragmatics*. Longman Group, Essex, UK, 1983.

[81] K. Bach, "Pragmatics and the philosophy of language," *The handbook of pragmatics*, vol. 463, p. 487, 2004.

[82] W. Bublitz and N. R. Norrick, *Foundations of pragmatics*, vol. 1. Walter de Gruyter, 2011.

[83] W. F. Hanks, *Deixis and indexicality*, pp. 315–346. New York, NY: Walter de Gruyter, 2011.

[84] M. Schwarz-Friesel and M. Consten, *Reference and anaphora*, pp. 348–372. New York, NY: Walter de Gruyter, 2011.

[85] Y. Huang, *Types of inference: entailment, presupposition and implicature*, pp. 397–421. New York, NY: Walter de Gruyter, 2011.

[86] E. Collavin, *Speech acts*, pp. 373–395. New York, NY: Walter de Gruyter, 2011.

[87] H. P. Grice, "Meaning," *The philosophical review*, vol. 66, no. 3, pp. 377–388, 1957.

[88] J. R. Searle, *Speech acts: An essay in the philosophy of language*, vol. 626. Cambridge university press, 1969.

[89] D. R. Traum, "Speech acts for dialogue agents," in *Foundations of rational agency*, pp. 169–201, Springer, 1999.

[90] P. R. Cohen and C. R. Perrault, "Elements of a plan-based theory of speech acts," *Cognitive science*, vol. 3, no. 3, pp. 177–212, 1979.

[91] C. R. Perrault and J. F. Allen, "A plan-based analysis of indirect speech act," *American Journal of Computational Linguistics*, vol. 6, no. 3-4, pp. 167–182, 1980.

[92] M. D. Sadek, "A study in the logic of intention.," *Proceedings of Knowledge Representation and Reasoning*, vol. 92, pp. 462–473, 1992.

[93] R. Vieira, Á. F. Moreira, M. Wooldridge, and R. H. Bordini, "On the formal semantics of speech-act based communication in an agent-oriented programming language," *Journal of Artificial Intelligence Research*, vol. 29, pp. 221–267, 2007.

[94] V. Raheja and J. Tetreault, "Dialogue act classification with context-aware self-attention," *arXiv preprint arXiv:1904.02594*, 2019.

[95] J. F. Allen, D. K. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent, "Toward conversational human-computer interaction," *AI magazine*, vol. 22, no. 4, pp. 27–27, 2001.

[96] B. De Carolis and G. Cozzolongo, "Interpretation of user's feedback in human-robot interaction," 2009.

[97] T. Williams, D. Thames, J. Novakoff, and M. Scheutz, "" thank you for sharing that interesting fact!" effects of capability and context on indirect speech act use in task-based human-robot dialogue," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 298–306, 2018.

[98] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[99] R. B. Miller, "Response time in man-computer conversational transactions," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pp. 267–277, 1968.

[100] J. L. Guynes, "Impact of system response time on state anxiety," *Communications of the ACM*, vol. 31, no. 3, pp. 342–347, 1988.

[101] T. Shiwa, T. Kanda, M. Imai, H. Ishiguro, and N. Hagita, "How quickly should communication robots respond?," in *2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 153–160, IEEE, 2008.

[102] E. Deng, B. Mutlu, and M. Mataric, "Embodiment in socially interactive robots," *arXiv preprint arXiv:1912.00312*, 2019.

[103] P. Wagner, Z. Malisz, and S. Kopp, "Gesture and speech in interaction: An overview," 2014.

[104] F. B. Mandal, "Nonverbal communication in humans," *Journal of human behavior in the social environment*, vol. 24, no. 4, pp. 417–421, 2014.

[105] D. Phutela, "The importance of non-verbal communication," *IUP Journal of Soft Skills*, vol. 9, no. 4, p. 43, 2015.

[106] C. P. Zeki, "The importance of non-verbal communication in classroom management," *Procedia-Social and Behavioral Sciences*, vol. 1, no. 1, pp. 1443–1449, 2009.

[107] H. Wang, "Nonverbal communication and the effect on interpersonal communication," *Asian Social Science*, vol. 5, no. 11, pp. 155–159, 2009.

[108] R. M. Sabatelli and M. Rubin, "Nonverbal expressiveness and physical attractiveness as mediators of interpersonal perceptions," *Journal of Nonverbal Behavior*, vol. 10, no. 2, pp. 120–133, 1986.

[109] S. R. Strong, R. G. Taylor, J. C. Bratton, and R. G. Loper, "Nonverbal behavior and perceived counselor characteristics.," *Journal of Counseling Psychology*, vol. 18, no. 6, p. 554, 1971.

[110] M. A. Bayes, "Behavioral cues of interpersonal warmth.," *Journal of Consulting and clinical Psychology*, vol. 39, no. 2, p. 333, 1972.

[111] R. Gomez, D. Szapiro, L. Merino, and K. Nakamura, "A holistic approach in designing tabletop robot's expressivity," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1970–1976, IEEE, 2020.

[112] T. Kucherenko, P. Jonell, S. van Waveren, G. E. Henter, S. Alexandersson, I. Leite, and H. Kjellström, "Gesticulator: A framework for semantically-aware speech-driven gesture generation," in *Proceedings of the 2020 International Conference on Multimodal Interaction*, pp. 242–250, 2020.

[113] Y. Yoon, B. Cha, J.-H. Lee, M. Jang, J. Lee, J. Kim, and G. Lee, "Speech gesture generation from the trimodal context of text, audio, and speaker identity," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–16, 2020.

[114] R. Buck and C. A. VanLear, "Verbal and nonverbal communication: Distinguishing symbolic, spontaneous, and pseudo-spontaneous nonverbal behavior," *Journal of communication*, vol. 52, no. 3, pp. 522–541, 2002.

[115] J. Xu, J. Broekens, K. Hindriks, and M. Neerincx, "Mood expression through parameterized functional behavior of robots," 08 2013.

[116] M. Suguitan, R. Gomez, and G. Hoffman, "Moveae: Modifying affective robot movements using classifying variational autoencoders," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 481–489, 2020.

[117] S. Kopp, B. Krenn, S. Marsella, A. N. Marshall, C. Pelachaud, H. Pirker, K. R. Thórisson, and H. Vilhjálmsson, "Towards a common framework for multimodal generation: The behavior markup language," in *International workshop on intelligent virtual agents*, pp. 205–217, Springer, 2006.

[118] L. Q. Anh and C. Pelachaud, "Expressive gesture model for humanoid robot," in *International Conference on Affective Computing and Intelligent Interaction*, pp. 224–231, Springer, 2011.

[119] C. Pelachaud, R. Gelin, J.-C. Martin, and Q. A. Le, "Expressive gestures displayed by a humanoid robot during a storytelling application," *New frontiers in human-robot interaction (AISB), Leicester, GB*, 2010.

[120] H. Van Welbergen, D. Reidsma, and S. Kopp, "An incremental multimodal realizer for behavior co-articulation and coordination," in *International Conference on Intelligent Virtual Agents*, pp. 175–188, Springer, 2012.

[121] T. Ribeiro, A. Pereira, E. Di Tullio, and A. Paiva, "The sera ecosystem: Socially expressive robotics architecture for autonomous human-robot interaction," in *2016 AAAI Spring Symposium Series*, 2016.

[122] R. Meena, K. Jokinen, and G. Wilcock, "Integration of gestures and speech in human-robot interaction," in *2012 IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 673–678, IEEE, 2012.

[123] M. Salem, S. Kopp, and F. Joublin, "Closing the loop: Towards tightly synchronized robot gesture and speech," in *International Conference on Social Robotics*, pp. 381–391, Springer, 2013.

[124] J. Hemminahaus and S. Kopp, "Towards adaptive social behavior generation for assistive robots using reinforcement learning," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pp. 332–340, IEEE, 2017.

[125] B. Ravenet, C. Pelachaud, C. Clavel, and S. Marsella, "Automating the production of communicative gestures in embodied characters," *Frontiers in psychology*, vol. 9, p. 1144, 2018.

[126] E. Balit, D. Vaufreydaz, and P. Reignier, "Pear: Prototyping expressive animated robots-a framework for social robot prototyping," in *HUCAPP 2018-2nd International Conference on Human Computer Interaction Theory and Applications*, p. 1, 2018.

[127] J. Hoberman, "Disney animation: The illusion of life," *Film Comment*, vol. 18, no. 1, p. 67, 1982.

[128] F. Bertacchini, E. Bilotta, and P. Pantano, "Shopping with a robotic companion," *Computers in Human Behavior*, vol. 77, pp. 382–395, 2017.

[129] E. Pot, J. Monceaux, R. Gelin, and B. Maisonnier, "Choregraphe: a graphical tool for humanoid robot programming," in *RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 46–51, IEEE, 2009.

[130] G. Van de Perre, H.-L. Cao, A. De Beir, P. G. Esteban, D. Lefeber, and B. Vanderborght, "Generic method for generating blended gestures and affective functional behaviors for social robots," *Autonomous Robots*, vol. 42, no. 3, pp. 569–580, 2018.

[131] N. Churamani, P. Barros, E. Strahl, and S. Wermter, "Learning empathy-driven emotion expressions using affective modulations," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2018.

[132] R. Desai, F. Anderson, J. Matejka, S. Coros, J. McCann, G. Fitzmaurice, and T. Grossman, "Geppetto: Enabling semantic design of expressive robot behaviors," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2019.

[133] G. Mier, F. Caballero, K. Nakamura, L. Merino, and R. Gomez, "Generation of expressive motions for a tabletop robot interpolating from hand-made animations," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1–7, IEEE, 2019.

[134] R. Plutchik, "A general psychoevolutionary theory of emotion," in *Theories of emotion*, pp. 3–33, Elsevier, 1980.

[135] M. Bear, B. Connors, and M. A. Paradiso, *Neuroscience: Exploring the brain*. Wolters Kluwer, 2015.

[136] J. R. Saffran, R. N. Aslin, and E. L. Newport, "Statistical learning by 8-month-old infants," *Science*, vol. 274, no. 5294, pp. 1926–1928, 1996.

[137] H. McGurk and J. MacDonald, "Hearing lips and seeing voices," *Nature*, vol. 264, no. 5588, pp. 746–748, 1976.

[138] T. Chen and R. R. Rao, "Audio-visual integration in multimodal communication," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 837–852, 1998.

[139] C. Darwin, *The expression of the emotions in man and animals*. John Murray, 1872.

[140] J. K. Burgoon and A. E. Bacue, "Nonverbal communication skills.," 2003.

[141] R. L. Birdwhistell, "Background to kinesics," *ETC: A review of general semantics*, pp. 10–18, 1955.

[142] J. S. Philpott, "The relative contribution to meaning of verbal and nonverbal channels of communication: A meta-analysis," *Unpublished master's thesis, University of Nebraska, Lincoln*, 1983.

[143] G. Calbris, *Elements of meaning in gesture*, vol. 5. John Benjamins Publishing, 2011.

[144] N. Ambady and R. Rosenthal, "Nonverbal communication," *Encyclopedia of mental health*, vol. 2, pp. 775–782, 1998.

[145] J. A. Harrigan, "Proxemics, kinesics, and gaze.," 2005.

[146] W. E. Rinn, "The neuropsychology of facial expression: a review of the neurological and psychological mechanisms for producing facial expressions.," *Psychological bulletin*, vol. 95, no. 1, p. 52, 1984.

[147] P. Ekman, "Facial expressions," *Handbook of cognition and emotion*, vol. 16, no. 301, p. e320, 1999.

[148] G. L. Trager, "Paralanguage: A first approximation," *Studies in linguistics*, vol. 13, pp. 1–11, 1958.

[149] B. Schuller, S. Steidl, A. Batliner, F. Burkhardt, L. Devillers, C. MüLler, and S. Narayanan, "Paralinguistics in speech and language—state-of-the-art and the challenge," *Computer Speech & Language*, vol. 27, no. 1, pp. 4–39, 2013.

[150] M. Karpiński, "The boundaries of language: Dealing with paralinguistic features," *Lingua Posnaniensis*, vol. 54, no. 2, pp. 37–54, 2012.

[151] U. Hess and P. Thibault, "Darwin and emotion expression.," *American Psychologist*, vol. 64, no. 2, p. 120, 2009.

[152] P. Ekman, "Are there basic emotions?," 1992.

[153] P. Ekman, "Methods for measuring facial action," *Handbook of methods in nonverbal behavior research*, pp. 45–90, 1982.

[154] B. De Gelder, "Why bodies? twelve reasons for including bodily expressions in affective neuroscience," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 364, no. 1535, pp. 3475–3484, 2009.

[155] A. Kleinsmith and N. Bianchi-Berthouze, "Affective body expression perception and recognition: A survey," *IEEE Transactions on Affective Computing*, vol. 4, no. 1, pp. 15–33, 2013.

[156] O. Da Pos and P. Green-Armytage, "Facial expressions, colours and basic emotions," *Colour: design & creativity*, vol. 1, no. 1, p. 2, 2007.

[157] P. Valdez and A. Mehrabian, "Effects of color on emotions.," *Journal of experimental psychology: General*, vol. 123, no. 4, p. 394, 1994.

[158] T. M. Sutton and J. Altarriba, "Color associations to emotion and emotion-laden words: A collection of norms for stimulus construction and selection," *Behavior research methods*, vol. 48, no. 2, pp. 686–728, 2016.

[159] P. Schillinger, S. Kohlbrecher, and O. von Stryk, "Human-robot collaborative high-level control with an application to rescue robotics," in *IEEE International Conference on Robotics and Automation*, (Stockholm, Sweden), May 2016.

[160] R. J. Kosinski, "A literature review on reaction time," *Clemson University*, vol. 10, no. 1, 2008.

[161] C. M. Carpinella, A. B. Wyman, M. A. Perez, and S. J. Stroessner, "The robotic social attributes scale (rosas) development and validation," in *Proceedings of the 2017 ACM/IEEE International Conference on human-robot interaction*, pp. 254–262, 2017.

[162] J. E. Cahn, "The generation of affect in synthesized speech," *Journal of the American Voice I/O Society*, vol. 8, no. 1, pp. 1–1, 1990.

[163] K. R. Scherer, "Expression of emotion in voice and music," *Journal of voice*, vol. 9, no. 3, pp. 235–248, 1995.

[164] A. Kappas, U. Hess, and K. R. Scherer, "Voice and emotion," *Fundamentals of nonverbal behavior*, vol. 200, 1991.

[165] R. F. Baumeister and M. R. Leary, "The need to belong: desire for interpersonal attachments as a fundamental human motivation.," *Psychological bulletin*, vol. 117, no. 3, p. 497, 1995.

[166] K. E. Powers, A. L. Worsham, J. B. Freeman, T. Wheatley, and T. F. Heatherton, "Social connection modulates perceptions of animacy," *Psychological science*, vol. 25, no. 10, pp. 1943–1948, 2014.

[167] S. R. Kellert and E. O. Wilson, *The Biophilia Hypothesis*. Shearwater, 1993.

[168] P. H. Kahn, "Developmental psychology and the biophilia hypothesis: Children's affiliation with nature," *Developmental Review*, vol. 17, no. 1, pp. 1 – 61, 1997.

[169] R. Ulrich and O. Lunden, "Effects of nature and abstract pictures on patients recovering from open heart surgery," in *International Congress of Behavioral Medicine*, jun 1990.

[170] Y. Joye and A. De Block, "'nature and i are two': A critical examination of the biophilia hypothesis," *Environmental Values*, pp. 189–215, 2011.

[171] t. e. Collins English Dictionary, "animacy." `https://www.thefreedictionary.com/animacy`, 2014. Accessed: 26-04-2021.

[172] S. Thorat, D. Proklova, and M. V. Peelen, "The nature of the animacy organization in human ventral temporal cortex," *Elife*, vol. 8, p. e47142, 2019.

[173] R. Gelman and E. Spelke, "2 the development of thoughts about animate and inanimate objects: implications for," *Social cognitive development: Frontiers and possible futures*, vol. 1, p. 43, 1981.

[174] G. Csibra, G. Gergely, S. Bíró, O. Koós, and M. Brockbank, "Goal attribution without agency cues: the perception of "pure reason" in infancy," *Cognition*, vol. 72, no. 3, pp. 237–267, 1999.

[175] A. Bugaiska, L. Grégoire, A.-M. Camblats, M. Gelin, A. Méot, and P. Bonin, "Animacy and attentional processes: Evidence from the stroop task," *Quarterly Journal of Experimental Psychology*, vol. 72, no. 4, pp. 882–889, 2019.

[176] A. S. Dick, S. Goldin-Meadow, U. Hasson, J. I. Skipper, and S. L. Small, "Co-speech gestures influence neural activity in brain regions associated with processing semantic information," *Human brain mapping*, vol. 30, no. 11, pp. 3509–3526, 2009.

[177] S. Kita, A. Özyürek, S. Allen, A. Brown, R. Furman, and T. Ishizuka, "Relations between syntactic encoding and co-speech gestures: Implications for a model of speech and gesture production," *Language and cognitive processes*, vol. 22, no. 8, pp. 1212–1236, 2007.

[178] M. Chu and S. Kita, "Co-thought and co-speech gestures are generated by the same action generation process.," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 42, no. 2, p. 257, 2016.

[179] D. McNeill, "Speech and gesture integration," *New Directions for Child Development*, pp. 11–28, 1998.

[180] J. P. De Ruiter, "14 the production of gesture and speech," *Language and gesture*, no. 2, p. 284, 2000.

[181] B. Butterworth and U. Hadar, "Gesture, speech, and computational stages: A reply to mcneill.," 1989.

[182] S. Kita, I. Van Gijn, and H. Van der Hulst, "Movement phases in signs and co-speech gestures, and their transcription by human coders," in *International Gesture Workshop*, pp. 23–35, Springer, 1997.

[183] N. S. Santos, N. David, G. Bente, and K. Vogeley, "Parametric induction of animacy experience," *Consciousness and cognition*, vol. 17, no. 2, pp. 425–437, 2008.

[184] D. H. Chang and N. F. Troje, "Perception of animacy and direction from local biological motion signals," *Journal of Vision*, vol. 8, no. 5, pp. 3–3, 2008.

[185] P. D. Tremoulet and J. Feldman, "Perception of animacy from the motion of a single object," *Perception*, vol. 29, no. 8, pp. 943–951, 2000.

[186] C. E. Looser and T. Wheatley, "The tipping point of animacy: How, when, and where we perceive life in a face," *Psychological science*, vol. 21, no. 12, pp. 1854–1862, 2010.

[187] K. Koldewyn, P. Hanus, and B. Balas, "Visual adaptation of the perception of "life": Animacy is a basic perceptual dimension of faces," *Psychonomic bulletin & review*, vol. 21, no. 4, pp. 969–975, 2014.

[188] B. Balas and C. Tonsager, "Face animacy is not all in the eyes: Evidence from contrast chimeras," *Perception*, vol. 43, no. 5, pp. 355–367, 2014.

[189] A. M. Rosenthal-von der Pütten, N. C. Krämer, and J. Herrmann, "The effects of humanlike and robot-specific affective nonverbal behavior on perception, emotion, and behavior," *International Journal of Social Robotics*, vol. 10, no. 5, pp. 569–582, 2018.

[190] C.-C. Chiu and S. Marsella, "Gesture generation with low-dimensional embeddings," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 781–788, 2014.

[191] C.-C. Chiu, L.-P. Morency, and S. Marsella, "Predicting co-verbal gestures: a deep and temporal modeling approach," in *International Conference on Intelligent Virtual Agents*, pp. 152–166, Springer, 2015.

[192] C. T. Ishi, D. Machiyashiki, R. Mikata, and H. Ishiguro, "A speech-driven hand gesture generation method and evaluation in android robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3757–3764, 2018.

[193] D. Hasegawa, N. Kaneko, S. Shirakawa, H. Sakuta, and K. Sumi, "Evaluation of speech-to-gesture generation using bi-directional lstm network," in *Proceedings of the 18th International Conference on Intelligent Virtual Agents*, pp. 79–86, 2018.

[194] T. Kucherenko, D. Hasegawa, G. E. Henter, N. Kaneko, and H. Kjellström, "Analyzing input and output representations for speech-driven gesture generation," in *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents*, pp. 97–104, 2019.

[195] Y. Yoon, W.-R. Ko, M. Jang, J. Lee, J. Kim, and G. Lee, "Robots learn social skills: End-to-end learning of co-speech gesture generation for humanoid robots," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 4303–4309, IEEE, 2019.

[196] L. Pérez-Mayos, M. Farrús, and J. Adell, "Part-of-speech and prosody-based approaches for robot speech and gesture synchronization," *Journal of Intelligent & Robotic Systems*, pp. 1–11, 2019.

[197] N. Sadoughi and C. Busso, "Speech-driven animation with meaningful behaviors," *Speech Communication*, vol. 110, pp. 90–100, 2019.

[198] A. Aly and A. Tapus, "Towards an intelligent system for generating an adapted verbal and nonverbal combined behavior in human–robot interaction," *Autonomous Robots*, vol. 40, no. 2, pp. 193–209, 2016.

[199] S. Ginosar, A. Bar, G. Kohavi, C. Chan, A. Owens, and J. Malik, "Learning individual styles of conversational gesture," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3497–3506, 2019.

[200] C. Ahuja, D. W. Lee, Y. I. Nakano, and L.-P. Morency, "Style transfer for co-speech gesture animation: A multi-speaker conditional-mixture approach," in *European Conference on Computer Vision*, pp. 248–265, Springer, 2020.

[201] B. Lugrin, J. Frommel, and E. André, "Combining a data-driven and a theory-based approach to generate culture-dependent behaviours for virtual characters," in *Advances in Culturally-Aware Intelligent Systems and in Cross-Cultural Psychological Studies*, pp. 111–142, Springer, 2018.

[202] B. Ghosh, A. Dhall, and E. Singla, "Automatic speech-gesture mapping and engagement evaluation in human robot interaction," in *2019 28th IEEE international conference on robot and human interactive communication (RO-MAN)*, pp. 1–7, IEEE, 2019.

[203] "Autonomous behaviour planning for socially-assistive robots in therapy and education," 2020.

[204] C. Ahuja and L.-P. Morency, "Language2pose: Natural language grounded pose forecasting," in *2019 International Conference on 3D Vision (3DV)*, pp. 719–728, IEEE, 2019.

[205] Y. Ferstl, M. Neff, and R. McDonnell, "Adversarial gesture generation with realistic gesture phasing," *Computers & Graphics*, vol. 89, pp. 117–130, 2020.

[206] C. Yu and A. Tapus, "Srg 3: Speech-driven robot gesture generation with gan," in *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 759–766, IEEE, 2020.

[207] R. Panchendrarajan and A. Amaresan, "Bidirectional lstm-crf for named entity recognition.," in *PACLIC*, 2018.

[208] H. Kumar, A. Agarwal, R. Dasgupta, and S. Joshi, "Dialogue act sequence labeling using hierarchical encoder with crf," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[209] R. Alzaidy, C. Caragea, and C. L. Giles, "Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents," in *The World Wide Web Conference*, WWW '19, (New York, NY, USA), p. 2551–2557, Association for Computing Machinery, 2019.

[210] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer, "Allennlp: A deep semantic natural language processing platform," *arXiv preprint arXiv:1803.07640*, 2018.

[211] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[212] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[213] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[214] C. Danescu-Niculescu-Mizil and L. Lee, "Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs.," in *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

[215] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, 2001.

[216] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[217] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[218] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[219] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[220] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[221] C. Sutton and A. McCallum, "An introduction to conditional random fields," *Foundations and Trends in Machine Learning*, vol. 4, no. 4, pp. 267–373, 2011.