

Autonomous Decision-Making for Socially Interactive Robots



Raúl Pérula-Martínez

Department of Systems Engineering and Automation
University Carlos III of Madrid

This dissertation is submitted for the degree of
Doctor of Philosophy

July 2017

TESIS DOCTORAL (DOCTORAL THESIS)

AUTONOMOUS DECISION-MAKING FOR SOCIALLY INTERACTIVE
ROBOTS

Autor (Candidate): Raúl Pérula-Martínez

Tutor (Adviser): Prof. Miguel A. Salichs

Director (Adviser): Dr. Álvaro Castro

Codirector (Adviser): Dr. Mohamed Abderrahim
(Universidad Carlos III de Madrid)

Tribunal (Review Committee)

Firma (Signature)

Chair: Dr. Fabio Bonsignorio

Member: Dr. Dolores Blanco

Secretary: Dr. Martin Stoelen

Título (Degree): Doctorado en Ingeniería Eléctrica, Electrónica y Automática

Calificación (Grade): _____

Leganés, de de

To those who really know they supported me.

The best preparation for tomorrow is
doing your best today.

H. Jackson Brown, Jr.

Acknowledgements

Ha sido un largo camino hasta llegar aquí. Ya me lo han dicho en varias ocasiones en el transcurso de la tesis: “hacer una tesis es como subir en una montaña rusa, subes, bajas, pero en todo momento tienes el corazón encogido”. Y que razón había.

De todos modos, al escribir estas palabras solo puedo pensar en agradecer a las personas que se han subido conmigo en la montaña rusa, que han vivido las bajadas donde el buen humor y los trabajos que salen a la primera hacen que quieras seguir montando un rato más; y en las subidas, cuando ves lo que te puede esperar después pero la angustia te inunda y no sabes qué va a pasar.

Antes de nada, agradecer a mi tutor, Miguel Ángel, la oportunidad que me brindó al aceptarme en el grupo de Robots Sociales y a mi director, Álvaro, por estar en todo momento ahí para ayudarme, aconsejarme, y apoyarme cuando lo he necesitado. Por supuesto, no me olvido de María, que aunque no aparezca en los créditos siempre ha estado presente poniendo su granito de arena. Gracias a los tres por dejarme continuar con vuestro trabajo, espero haber correspondido de igual manera.

No puedo olvidarme tampoco de dos grandes personas a las cuales respeto y admiro, que me tendieron una mano cuando más lo necesité. Gracias José María y Carlos por ser como sois. Y por supuesto agradecer a mi co-director Mohamed y al resto del RoboticsLab por ofrecerme la oportunidad de vivir esta etapa en la universidad.

Por supuesto, esta aventura no habría sido posible sin mis compañeros de laboratorio. Los chicos hardware David, Iñigo y Marcos. Los chicos software Alberto y Jony. Y el resto de compañeros que se han ganado toda mi admiración, Víctor, José Carlos, Fernando Alonso, Javi, Irene, Esther y un largo elenco de alumnos de grado y máster que nos han dejado muy buenos momentos.

También dar unas gracias especiales a dos grandes compañeros de batalla, Juanmi y Pablo, grandes personas y buenos amigos. Y que decir sobre mis socios, amigos y compañeros Félix y Vero. Gracias a vosotros por la nueva aventura que comenzamos juntos sacando adelante CREA Robótica Educativa.

Por último, y por ello lo más importante, mi familia. Agradecer a mi madre todo el esfuerzo que hizo para que hoy yo esté aquí cumpliendo mis sueños. Eres lo mejor

que tengo en este mundo. Y que decir sobre Esther, mi compañera de fatigas en el día a día que me siempre me ha ofrecido su mejor cara en los buenos y no tan buenos momentos. Por supuesto, no puedo dejarme a mi padre y mis hermanos, gracias por estar ahí.

Una vez más, gracias a todos por haber disfrutado de este viaje en la montaña rusa de mi tesis.

Abstract

The aim of this thesis is to present a novel decision-making system based on bio-inspired concepts to decide the actions to make during the interaction between humans and robots. We use concepts from nature to make the robot may behave analogously to a living being for a better acceptance by people. The system is applied to autonomous Socially Interactive Robots that works in environments with users. These objectives are motivated by the need of having robots collaborating, entertaining or helping in educational tasks for real situations with children or elder people where the robot has to behave socially. Moreover, the decision-making system can be integrated into this kind of robots in order to learn how to act depending on the user profile the robot is interacting with. The decision-making system proposed in this thesis is a solution to all these issues in addition to a complement for interactive learning in HRI. We also show real applications of the system proposed applying it in an educational scenario, a situation where the robot can learn and interact with different kinds of people. The last goal of this thesis is to develop a robotic architecture that is able to learn how to behave in different contexts where humans and robots coexist. For that purpose, we design a modular and portable robotic architecture that is included in several robots. Including well-known software engineering techniques together with innovative agile software development procedures that produces an easily extensible architecture.

Resumen

El objetivo de esta tesis es presentar un novedoso sistema de toma de decisiones basado en conceptos bioinspirados para decidir las acciones a realizar durante la interacción entre personas y robots. Usamos conceptos de la narutaleza para hacer que el robot pueda comportarse análogamente a un ser vivo para una mejor aceptación por las personas. El sistema está desarrollado para que se pueda aplicar a los llamados Robots Socialmente Interactivos que están destinados a entornos con usuarios. Estos objetivos están motivados por la necesidad de tener robots en tareas de colaboración, entretenimiento o en educación en situaciones reales con niños o personas mayores en las cuales el robot debe comportarse siguiendo las normas sociales. Además, el sistema de toma de decisiones es integrado en estos tipos de robots con el fin de que pueda aprender a actuar dependiendo del perfil de usuario con el que el robot está interactuando. El sistema de toma de decisiones que proponemos en esta tesis es una solución a todos estos desafíos además de un complemento para el aprendizaje interactivo en la interacción humano-robot. También mostramos aplicaciones reales del sistema propuesto aplicándolo en un escenario educativo, una situación en la que el robot puede aprender e interaccionar con diferentes tipos de personas. El último objetivo de esta tesis es desarrollar una arquitectura robótica que sea capaz de aprender a comportarse en diferentes contextos donde las personas y los robots coexistan. Con ese propósito, diseñamos una arquitectura robótica modular y portable que está incluida en varios robots. Incluyendo técnicas bien conocidas de ingeniería del software junto con procedimientos innovadores de desarrollo de software ágil que producen una arquitectura fácilmente extensible.

Contents

List of Figures	xvii
List of Tables	xix
Nomenclature	xxii
1 Introduction and Overview	1
1.1 Motivation	1
1.1.1 Socially Interactive Robots	2
1.2 Open Challenges	5
1.3 Objectives	7
1.4 Organization of the document	9
2 State of the Art	11
2.1 Introduction	11
2.2 Bio-inspired decision-making in robotics	11
2.3 Decision-Making for HRI	19
2.4 Conclusion	20
3 Autonomous Decision-Making	21
3.1 Introduction	21
3.2 Previous works	21
3.3 The Decision-Making process	24
3.4 The state of the robot	26
3.4.1 External states	27
3.4.2 Internal state	28
3.5 Selecting the action	29
3.6 Acting in the world	31
3.6.1 Skills	31

Contents

3.6.2	Memories	35
3.7	Learning how to act	38
3.8	Conclusion	41
4	The Robotic Architecture	43
4.1	Introduction	43
4.2	Previous works	44
4.3	The Software Architecture	46
4.3.1	Software Methodologies Included	48
4.3.1.1	Design Patterns	48
4.3.1.2	Agile Software Development	52
4.3.2	Robot Operating System	56
4.3.3	Implementing the low-level components	59
4.3.4	Implementing the high-level components	64
4.4	The Decision-Making System	68
4.4.1	The States Managers	70
4.4.2	The DMS	73
4.4.3	The Action Manager	74
4.5	Conclusion	77
5	Experimental Results	79
5.1	Introduction	79
5.2	Our Socially Interactive Robots	79
5.2.1	Maggie	80
5.2.2	Mini	82
5.3	Experimental Design	83
5.3.1	Scenario description	84
5.3.2	Hypotheses	84
5.3.3	Task	85
5.3.4	Implementation	86
5.3.5	Participants	92
5.4	Results	94
5.4.1	Effectiveness and Efficiency of the system	94
5.4.1.1	Learning by doing	94
5.4.1.2	Wellbeing	96
5.4.2	User's satisfaction	99
5.5	Discussion	100

5.6	Conclusion	101
6	Conclusions and Future Works	103
6.1	Conclusions	103
6.2	Summary of the Key Contributions	104
6.3	Future Works	105
6.4	List of Publications	106
6.4.1	Journals	106
6.4.2	Conferences and Workshops	107
6.4.3	Source Code	107
	References	109

List of Figures

2.1	Behavior system in three layers from behavioral psychology.	12
2.2	The Cathexis architecture.	14
2.3	The role of drives in behavior selection.	15
2.4	An overview of the emotion system for the Kismet robot.	16
2.5	Overview of the parts involved at behavioral level of EFSA.	17
2.6	Overview of the social behavior controller architecture.	18
3.1	Biological inspired decision-making system developed by Malfaz.	22
3.2	DMS components flowchart.	25
3.3	Available states for a user.	27
3.4	An example for several different drives.	29
3.5	Skills flowchart.	32
3.6	Interruption process flowchart.	34
3.7	Reinforcement Learning framework.	39
4.1	Original AD Control Architecture.	45
4.2	AD Control Architecture including the DMS.	46
4.3	General scheme for the software architecture in layers.	47
4.4	Interface Pattern	50
4.5	Model-View-Controller Pattern example.	52
4.6	Test-First Development flowchart.	53
4.7	ROS general scheme.	56
4.8	ROS communication between nodes.	57
4.9	The ActionClient and ActionServer communication.	58
4.10	UML class diagram for low-level components.	59
4.11	UML package diagram for Maggie and Mini devices.	61
4.12	UML class diagram for the labjack device.	62
4.13	General scheme for some skill nodes depending on devices.	65

List of Figures

4.14	UML package diagram for common skills.	66
4.15	Diagram for the touch sensor skill.	67
4.16	DMS components flowchart.	68
4.17	Database diagram for components.	69
4.18	UML class diagram modeling the internal and external states.	70
4.19	People state monitoring.	71
4.20	UML package diagram for the DMS.	73
4.21	Database diagram for users (episodic memory).	77
5.1	Social Robot Maggie.	80
5.2	Hardware architecture for the social robot Maggie.	81
5.3	Social Robot Mini.	82
5.4	Hardware architecture for the social robot Mini.	83
5.5	Positions for the user and the robot when interacting.	85
5.6	Sessions defined for the HRI experiments.	87
5.7	Evolution chart for the user's satisfaction drive.	88
5.8	Evolution chart for the rest drive.	88
5.9	Evolution chart for the interaction drive.	89
5.10	Wellbeing of the robot in the experiment with user profile 1.	97
5.11	Wellbeing of the robot in the experiment with user profile 2.	98

List of Tables

3.1	Memories definitions.	35
4.1	States stored in the database.	72
4.2	Actions stored in the database.	74
4.3	Actions effects stored in the database.	76
5.1	Motivations-Drives relation.	87
5.2	Motivations-External stimulus relation.	90
5.3	Repertoire of actions available for the DMS.	91
5.4	Actions-Effects relation.	92
5.5	Actions learned (maximum Q-Value) for the user profile 1.	94
5.6	Actions learned (maximum Q-Value) for the user profile 2.	96
5.7	Average and standard deviation for the wellbeing of the robot.	99
5.8	Percentage of time the robot has perceived the user is satisfied.	99

Nomenclature

Acronyms / Abbreviations

AD Automatic-Deliberative Architecture

AP Architectural Patterns

ASD Agile Software Development

DMS Decision-Making System

EM Episodic Memory

FDP Fundamental Design Patterns

IoC Inversion of Control

IoT Internet of Things

LOA Level Of Autonomy

LTM Long-Term Memory

MVC Model View Controller pattern

OOP Object Oriented Programming

PM Procedural Memory

RA Robotic Architecture

RL Reinforcement Learning

ROS Robot Operating System

SAR Socially Assistive Robots

Nomenclature

SIR Socially Interactive Robots

SM Semantic Memory

STEM Science, Technology, Engineering, and Mathematics

STM Short-Term Memory

TDD Test-Driven Development

TFD Test-First Development

WM Working Memory

Chapter 1

Introduction and Overview

The beginning is the most important part of the work.

Plato

This chapter introduces this thesis, its motivation, its challenges and objectives, and describes the organization of the manuscript.

1.1 Motivation

Robots have been gradually leaving laboratories and factories and moving into human populated environments. Either as companions, tutors, receptionists, or cleaners, such robots are needed to complement our quality of life. But the reality is that those robots are very limited at present. They need a standard and fully functional architecture to work uninterruptedly in a long-term interaction with people. This architecture needs to fulfill some specific features such as modularity, portability, and easy to upgrade, among others. Furthermore, there must be a high level layer of abstraction to manage every single action the robot can do with the input information obtained from the surrounding environment, this is the decision-making system.

In this way, there are some concerns related to our society. The aging of the population is one of the most relevant at this time. We are suffering severe changes because the global economy has evolved and families prefer to have fewer children than before. However, the world population (the total number of humans currently living) is estimated at 7.4 billion as of August 2016 [1]. The United Nations projects it to reach 8 billion in 2024 and 10 billion in the year 2056. As a consequence, people is becoming much more aged and a lot of them will need special care or more quality of life.

Leading powers in the world are relying on the future of robotics [2]. We are currently living the fourth industrial revolution. This is in part because robotics is changing industries and society. Nowadays, robotics can be applied in much more areas than traditional industry, for instance in education or health care. Indeed, educational robotics are growing to become a discipline included in schools. This is due to it involves several important topics such as mechanics, electronics, and computer science among others.

Nevertheless, autonomy in robotics is one of the outstanding issues that needs to be solved. Nowadays, most of the robots operating out of the factories can only work autonomously a short time period usually in a controlled environment. Real environments are changing continuously due to people actions or even because of the objects we can find in it. The most advanced robots in this age usually may only be in this kind of controlled area either doing a preprogrammed and sequential tasks.

Related to the robots' autonomy, it is also relevant the process of making decisions. Another open research field with no standardization. From humans, we know that a person makes a lot of erroneous decisions during his life. This is not bad actually, because we obtain new knowledge and make the next decisions based in our memorized experiences. Furthermore, we learn from these experiences new ways of doing the same actions or change totally our previous decisions to improve our repertoire of possible actions. In this way, we can simulate this kind of behaviors with bio-inspired systems that help robots to make decisions autonomously.

For that reason, we can find several open questions that need to be answered. For instance, how could a robot make some decisions to interact with people knowing about its real limitations such as following the social rules? Could this kind of robots be able to be fully autonomous? These are some real questions we try to answer in this thesis, at least, giving an initial solution in the field of **Socially Interactive Robots**.

1.1.1 Socially Interactive Robots

Continuing with the motivation of this thesis, it is important to clearly define what a Socially Interactive Robots (SIR) is. According to Fong [3]:

“A Socially Interactive Robot is an autonomous or semi-autonomous robot that interacts and communicates with humans by following the behavioral norms expected by the people with whom the robot is intended to interact.”

In the last 10 years, there has been a lot of studies related to how robots can be included in our society [4] [5] [6] [7] [8]. As commented previously, countries such as USA and Japan are investing great part of their budgets for including robotics to improve factories, education systems, and health care services in our everyday life.

Industrial robots are evolving to be more collaborative and not only a machine that do a repetitive task [9]. There is coming a new line of industrial collaborative robots that will be able to do more than a repetitive task in a controlled space, they are going to help people in their work spaces [10].

But industrialization is not the only area that is evolving. **Health care** [11] is also suffering changes to integrate robots for helping caregivers in cognitive and physical therapies. This can happen thanks to new morphologies applied through giving them a friendly appearance. Thus, the robots may interact with patients in a natural way. Most of time, interactions are based on typical sessions in which the robot usually supports the caregiver in some specific tasks.

Another important area that is getting more and more importance around robotics is **education** [12] [13] [14]. Either as robotic tutors or simply as multifunctional platforms to develop new features, educational robotics is getting followers around the world.

In spite of all these areas, we cannot forget the entertaining area because it is one of the most demanded. Robots will be able to tell us a story in a multimedia manner [15], check our mail or advise us about a meeting, or even play a soccer match [16]. This robots will be even able to dance or sing [17].

As it can be checked, robots are becoming to be not only a machine in a factory doing a specific task, but now will be faithful companions to help us in the most arduous or pleasant tasks [18].

Following, two important concepts related to SIR are presented: **autonomy** and **sociability**.

Autonomy is a feature that is a major aspect for SIR [19]. The lack of this property produce a large constrain in HRI usually not being able to interact in changing environments.

Related to this, there is a concept that can characterize the extent of human involvement in the interaction. This is the *Level Of Autonomy* (LOA) [9] [19]. The progressive scale for the LOA may be from “human does it all” to “computer decides everything and acts autonomously”.

Introduction and Overview

But, what could we expect for a Socially Interactive Robot? This kind of robots are pretended to be at least semi-autonomous, that means they can do several tasks autonomously but might learn new actions or knowledge from people.

Research about SIR is going to work, entertain and live together with fully autonomous robots in real environments, where the robot must be capable of performing long-term interactions. There are many examples of this kind of long-term interaction [20] [21] [22] [23], but all of them depend on the autonomy of the robot.

In this way, when we refer to autonomy for SIR the topic is related to a robot making actions by itself with no human control at all. Robots are commonly pre-programmed to do a single task. But if we want to include robots in our day-to-day life, robots have to be able to make decisions involving one or several tasks to achieve the final goal. Thus, the repertoire of actions a robot could do, should not be a limitation.

Consequently, long-term interaction scenarios help researchers to solve unknown issues that until now were not discovered in everyday life. This will provide the autonomy desire for the near future robots.

Joining all these capabilities could constitute the complete Socially Interactive Robot. That is what several researchers and companies are currently researching and trying to develop. They want to include robots in our lives as companions but working permanently. That means a robot working 365 days, 7 days a week, 24 hours a day.

On the other hand, *sociability* is a property that defines the human behavior when interacting between them following the society rules. Robots do not know a priori how they should behave when interacting with people.

There are many applications of social robots in daily situations. Next, some of them are described to understand how sociability is included.

Either people or robots behave differently depending on the social situation. Whether the interaction is with elder people the behavior must be respectful and polite in every case. However, interacting with children can change the situation, being possible to behave in a less formal manner. It also depends whether the person is known or unknown. We are usually more familiar with people we know from long time, even not direct family. Thus, people interacts continuously at work, at home or in a residence, at schools, etc. but their behavior will adapt to the situation.

Although most of the robots are often in technical laboratories, we can find examples in which robots are working in homes or public spaces. Even so, we can find studies that say that one of the most accepted robots interacting in homes are *cleaner robots* [24]. These robots are often working continuously in long-term interactions at home

[25]. Nevertheless, robotic toys are also accepted to interact with children in a natural way [26].

In a more professional environment, there are examples of robots handling situations in a office helping workers to carry on some packages or delivering the mail [27]. We can also find this kind of treatment in shopping malls [28], testing whether a polite or impolite behavior affects the way people purchase.

Now, we live in a more and more technological world where things are connected. Intelligent homes or cars, wearables, and smartphones use the Internet as a bridge to get information from people. Thus, we can find some studies that try to apply SIR in applications where the robot is connected to a house to extend independent living of patients [29]. In a near future, this kind of robots will be able to benefit these new technologies used in conjunction to improve the interaction with people. Being now almost a reality in examples of robots in domestic applications [30] [31].

Nevertheless, there are special situations that need special attention. This is the case of health care. Cognitive and physical rehabilitation usually is a great effort for caregivers that need complementary help for a specific session. Both rehabilitation and stimulation for elderly requires patient and dedication, more even when it is in home [32]. Thus, several studies reveals that neurological disease in elderly have a good acceptance when robots are used [33]. Moreover, it is demonstrated that children who suffers autism react better to sessions when technology is present, and even better when robots play a relevant role [34] [35].

It should be notice that this kind of robots is a special type of SIR called Socially Assistive Robots (SAR). They are usually focused on help caregivers in special sessions, both for physical or cognitive stimulation. Despite everything, it is demonstrated that people respond better to robots than a simple automated system or computer tablets in health care [36]. This studies will help to keep studying the effects of human-robot interaction in other fields (e.g. education or entertainment).

There are many other applications of these robots such as companions used by elderly people [37] [38] [39] [40] [41], or tutors in schools [42] demonstrating that SIR would become loyal partners in a near future.

1.2 Open Challenges

There are many open challenges in the field of Socially Interactive Robots. Some of the most important are addressed in this thesis:

Introduction and Overview

- **Autonomous robots.** Until now, robots still are limited on how they can interact with the world autonomously. Usually, when working in controlled spaces such as laboratories or industries, robots are preprogrammed to execute a series of tasks that normally cannot be applied to other unstructured environments. Therefore, autonomy is considered as a real challenge that has to be studied in full-time interaction. Robotic autonomy is an open challenge necessary to achieve for robots that work continuously interacting in continuous changing environments with people around it.
- **Decision-Making processes.** This is the main open challenge around this thesis. The way a robot can make decisions is a topic that is being studied at present but nobody knows exactly how to implement a standard methodology in robotics. Moreover, there are different levels of abstraction where decisions can be made. For instance, it is not the same to make a decision in a conversation, where the robot would have to decide what to say. Or making decisions about what action to execute (e.g. walking, waiting, or turning on a TV). That means that can be many levels in which modules such as dialog, movement, or sensor information act as inputs of the decision making system.
- **Environment monitoring.** There are a lot of objects and people in our surroundings. How the robot is able to detect and control the changes is key in robotics. Furthermore, to monitor people changes is one of the hardest challenges studied in perception and robotics. Even more relevant when the robot is making decisions receiving some feedback from its own actions executed. Thus, exogenous actions, actions performed by others not caused by an action done by oneself, has to be modeled and monitored in order to provide to the robot the way to know what is happening in its surrounding.
- **Human-Robot Interaction (HRI).** Collaboration with robots in health care or education is a practice that is becoming more and more used every day. Although robots are still used as passive tools or mainly teleoperated to facilitate the interaction. Little by little, robots are being used more often for example in houses or schools. In a near future, we will be able to find even humanoid robots cleaning our houses, washing the dishes, feeding our children, or taking care of our elderly in a nursing home. These are only some examples of how Socially Interactive Robots are helping us in our everyday tasks but the open challenge is to find the way to make them completely autonomous.

- **Learning by interaction.** There are many learning techniques depending on what the robot needs to learn. The robot should be able to get new knowledge from the surrounding environment – either objects or people. Reinforcement learning (RL) is a machine learning technique commonly used in robotics concerned with how they ought to take actions in an environment. The Q-learning technique is one kind of RL that can be used to find an optimal action-selection policy for any given *Markov decision process*. We can find several open challenges related with this topic but it is of interest of this thesis the related with how to include this kind of learning technique in conjunction with HRI.

With the inclusion of deep learning techniques, where the system can learn from lots of information, robotics will be able to benefit only in long-term interactions. That changes the way robotics traditionally work expanding the range of possibilities in which it could be included. Nevertheless, there is no previous knowledge of applying this kind of techniques in robotics due to the difficulty involved that kind of experiments.

- **Robotic Architectures.** All robot needs a complete architecture - both hardware and software, to be fully functional. Although hardware has been evolving continuously during years, control algorithms have been the real protagonists. Nevertheless, software is playing now an important role that cannot be omitted. Software engineering techniques are being applied every day more and more in robotics to improve their efficiency. Thus, robotics architectures are including this kind of techniques but there is still an open challenge of standardize the low and medium level to facilitate the decision making processes.

1.3 Objectives

This thesis aims to solve or, at least, get a forthcoming solution for the challenges stated in the previous section. The focus of this thesis is on the integration of a robotic architecture in addition to a bio-inspired Decision-Making System for Socially Interactive Robots.

From the challenges presented in the previous section, we can define the main objective of this thesis:

Introduction and Overview

*To develop a **Decision-Making System** integrated in a **robotic architecture** that allows an autonomous **Socially Interactive Robot** interacting with different kind of **users**.*

To achieve this objective, some research aspects need to be addressed:

1. To develop a Decision-Making System (DMS).
 - (a) *Developing a modular DMS based on bio-inspired concepts.*
 - (b) *Studying memory components to facilitate interaction and learning.*
 - (c) *Studying a reinforcement learning technique to learn from HRI.*
 - (d) *Controlling exogenous actions through external states.*
 - (e) *Testing the DMS is multi-platform working in different robots.*
2. To design the Robotic Architecture (RA).
 - (a) *Implementing all software components (drivers + devices + skills) to work in a standard framework.*
 - (b) *Applying software engineering techniques to standardize the way new components will be created in the architecture.*
 - (c) *Testing the complete system (HW + SW) and checking it is working properly in real robots.*
3. To evaluate the DMS in a HRI scenario.
 - (a) *Tunning the learning process to define a standard process of exploration and exploitation.*
 - (b) *Demonstrating that the system is able to be used in a HRI scenario with several types of users.*

All these objectives aim to be an initial step to demonstrate that decision-making is an essential process for Socially Interactive Robots. The novelty of this thesis is centered to standardize the robotic architecture used in addition to test the autonomous robot making its own decisions in a realistic scenario.

1.4 Organization of the document

This thesis is divided in six chapters, each one dedicated to different objectives explained before. Chapter by chapter, the contents are organized as follows:

Chapter 2 includes a review of the most relevant works related to the topics of this thesis. An overview about the decision-making systems in the fields of social robotics and human-robot interaction are presented. The state-of-the-art presents previous works and other alternatives about how to make decisions mainly in robotics.

Chapter 3 addresses the decision making system designed for Socially Interactive Robots. This is the main chapter of this thesis. For that reason, a complete description of the previous works is included in this chapter. Followed by the description of the decision-making process emphasizing each one of the innovative concepts included.

Chapter 4 describes the proposed robotic architecture and decision-making system developed for Socially Interactive Robots. In this chapter it is described the robotics platforms used in this thesis. It is also described the implementation of every component in the DMS.

Chapter 5 presents the evaluation of the proposed system. The results from the experiments are addressed showing the theoretical concepts applied in the DMS and tested in a HRI scenario with different kind of users.

Chapter 6 concludes this thesis giving the reader a general vision of the issues considered about Socially Interactive Robots. It summarizes the key contributions and the lines of research that could be tackle in a future.

Chapter 2

State of the Art

People who think they know
everything are great annoyance to
those of us who do.

Isaac Asimov

2.1 Introduction

This chapter addresses the state of the art of the main topics related to this thesis. First, some of the more relevant works related with decision-making in robotics are introduced emphasizing some of the biologically inspired decision-making processes developed in other previous works. Secondly, a general view about decision-making for human-robot interaction is also presented.

Additionally, we include some relevant previous works in Chapter 3 and 4 in order to give to the reader more information about those topics in particular.

2.2 Bio-inspired decision-making in robotics

Action selection is one the major fields of study in robotics from decades. Researchers has been studying the human being and other animals in order to know how we select the actions we perform. From biological studies [43], we know that there are two kind of behaviors: *innate* and *voluntary*. Innate behaviors are those that we do unintentionally as reflex. On the other hand, voluntary behaviors are determined from sensory inputs in a deliberative process.

Many authors get inspired by neurobiological principles to model social robot behaviors using neural networks (ANN) techniques [44] [45]. However, the main difficulty limiting the ANNs is that the learning process have to face to some issues in real-time learning when having continuous changing environments [46].

For that reason, the development of reasoning systems that can select a meaningful action in a continuously changing environment already was an important area of research in AI at the beginning of the 90s. In that moment, researchers looked for systems that support goal-directed tasks in addition to reactivity to unanticipated changes in the environment [47]. According to Brooks, he pretended to create robots able to interface in real environments through reactive perception [48]. Where reactive systems were able to select a specific action only depending on the changes of their surroundings.

On the contrary, researchers in behavioral psychology and artificial intelligence use to accept that a behavior system is organized in three layers (Fig. 2.1), where the nomenclature of each layer and its components might be diverse.

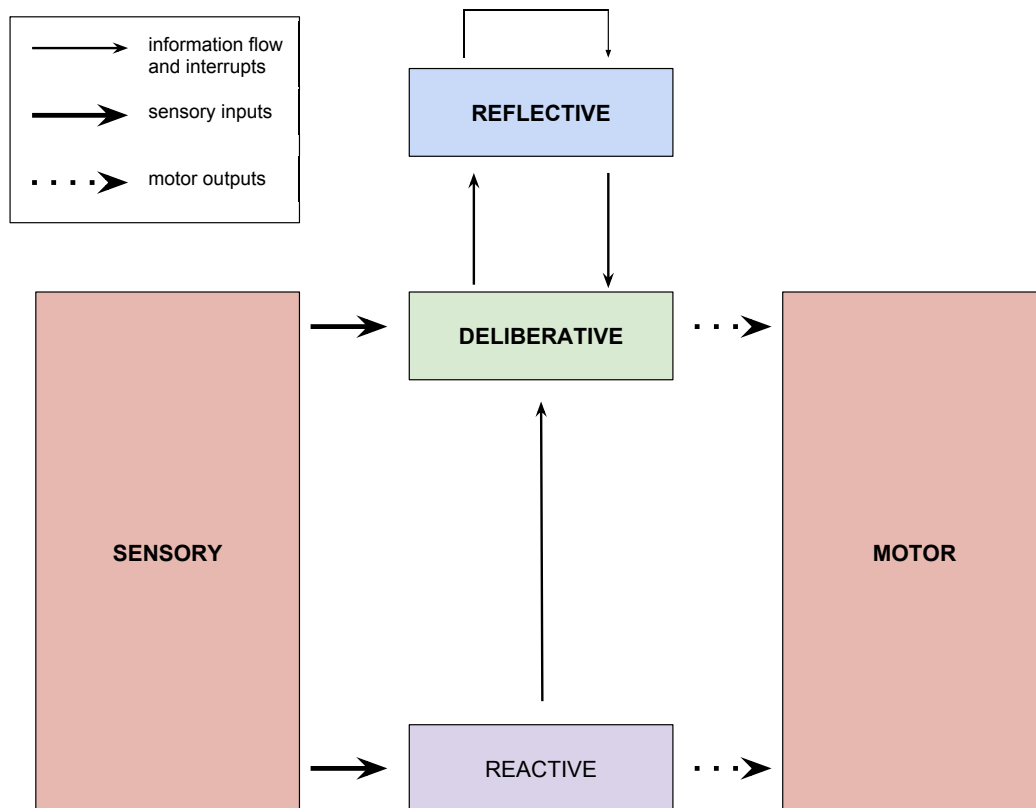


Fig. 2.1 Behavior system in three layers from behavioral psychology.

2.2 Bio-inspired decision-making in robotics

The terms of *reactive*, *deliberative*, and *reflective* layers comes from theories of different authors such as Orthony [49][50] and Sloman [51]. Where the *reactive* layer is constituted by the lowest level processes of sensorimotor stimuli which are generally determined. Information from sensor inputs is instantly acted upon with appropriate outputs. Usually, the behaviors related to this layer are compared to the innate behaviors. The *deliberative* layer corresponds to complex and skilled behaviors guiding most of the voluntary actions. At this layer, the agent is able to plan taking the information from the reactive layer and the reflective layer. The plan established is then executed to fulfill a goal. Lastly, the *reflective* layer is the most complex because it is produced the real deliberation of the actions. Dissimilar to the lower layers, the reflective layer is not connected to any low-level information. This layer evaluates the plans and decide how to better achieve the goal proposed.

But the integration of *planning* or *acting* functions is still a critical issue in the design of a deliberative system. Ingrand et al. [52] describe three design options depending on the the level of decision. If the *emphasis relies on planning*, possibly it manages nondeterministic models of the world that many times does not work in a continuously changing environment. But that guarantee of achieving the goal based on the inputs provided to a predictive model. Thus, the applicability of this idealistic view is limited the low predictability of most environments. Otherwise, if the *emphasis is on acting*, this has to handle nondeterministic usually partially observable and dynamic environments. But this option puts the most of the load in deliberation with the real world constraints (imperfect sensors and actuators) on acting.

Therefore, the best option seems to be a *balanced decomposition* where the major design option is how to better decompose the complex deliberation problem between planning and acting. A balanced decomposition should apply predictive capabilities to planning as well as to acting.

In this way, a robot uses a decision-making system to solve two main issues: “what to do” and “when to do”. Among various approaches, the *homeostatic drive* theory is one of the most studied [53]. According to Cannon [54], *homeostasis* means maintaining a stable internal state. Concretely, an agent has a number of internal needs such as hunger, companion, fun, etc. which have to be kept within certain ranges. When one or several needs are not satiated, a dominant motivation is created. This motivation provokes that an action is executed in order to receive a feedback that satiate that need.

There are several *homeostatic-based* architectures in the field of robotics that deserve special attention.

The first is the developed by Velasquez [55]. He proposed a distributed computational model which offered an alternative approach to model the dynamic nature of different affective phenomena providing a flexible way of modeling their influence on the behavior. The *Cathexis* model (Fig. 2.2), name that this architecture received, is an architecture based mainly in emotions that includes a behavior system. This system is a distributed system composed of a network of behaviors, such as “smile” or “express something”. Each of these behaviors are competing for the control of the agent. His behaviors contains two major components: the Expressive or Motor component and the Experiential component. The Expressive component contemplates aspects like prototypical facial expression, body posture, and vocal expression. On the other hand, the Experiential component includes Motivations that affect the levels of drives, and Action tendency and readiness that are modeled by behaviors. The selection of actions in this architecture is made by a competition among behaviors in order to obtain the control of the agent. At some point in the time, the behavior with highest value becomes the active behavior. This value is calculated from the motivations and a wide variety of external stimuli.

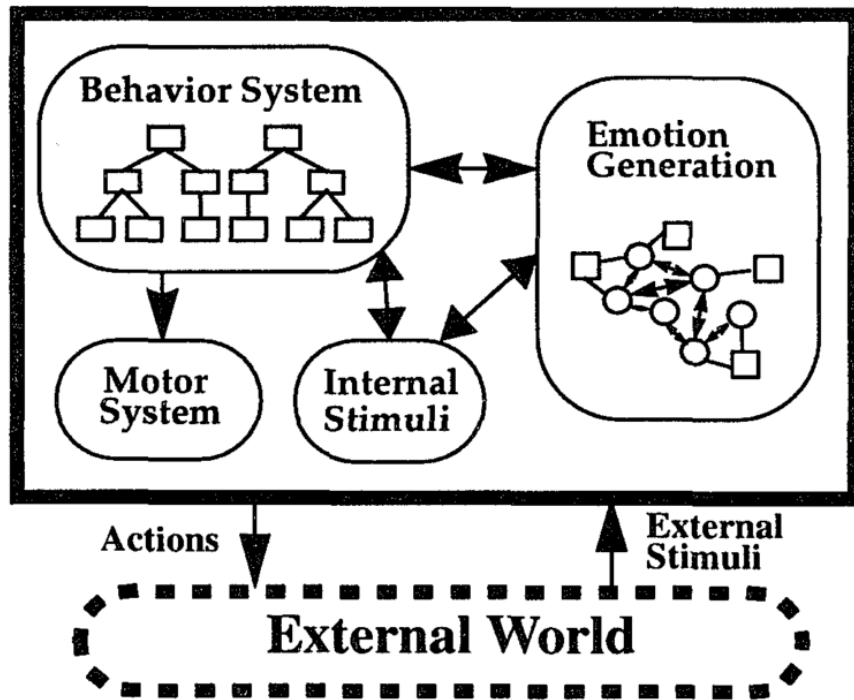


Fig. 2.2 The Cathexis architecture.

Following we can find the studies made by Arkin et al. [56] for the Sony’s entertainment robots AIBO. They studied the role of ethological and emotional

models as the basis for an architecture that includes a behavior system (Fig. 2.3). The mechanism of action selection included in this system is based on evaluating both external and ongoing internal drives. They employed the “homeostasis regulation rule” where internal variables are specified and maintained within proper ranges. Behavioral actions and changes withing the environment produce change in the internal variables. The rule for action selecting that they implemented was based then in using the regulation of the internal variables as a motivational “drive” signal for the behavior modules.

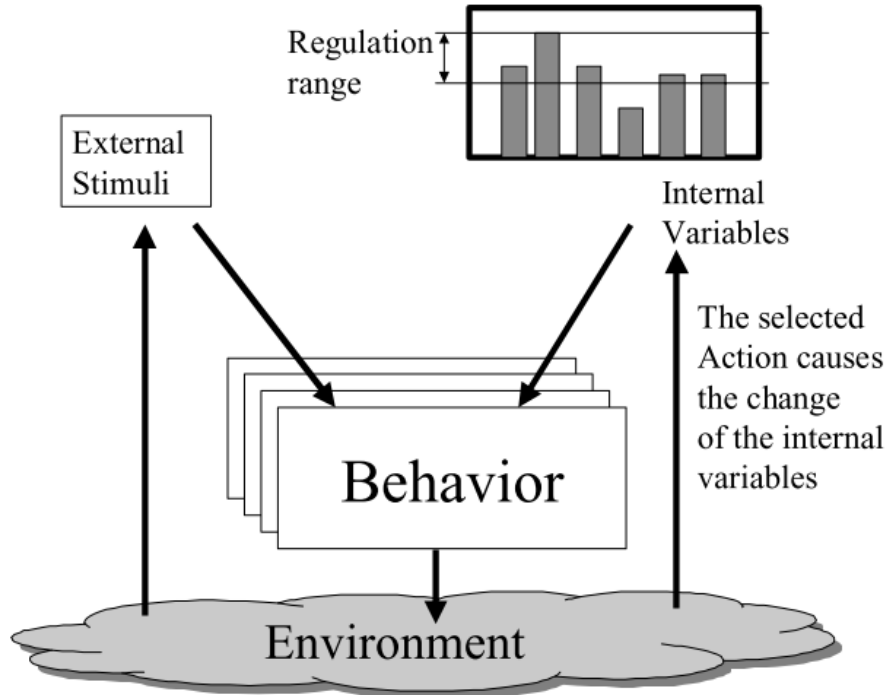


Fig. 2.3 The role of drives in behavior selection.

According to Cañamero [57], one of the main problems that motivationally autonomous agents are confronted with is *activity-action* is the *behavior-selection*. How to choose the appropriate behavior at each point in time so as to work towards the satisfaction of the current goal (it most urgent need). She comments that motivational states, such as “hunger”, “social attachment”, etc. are drives that constitute urges to action based on internal needs related with survival. The architecture she proposed [58][59], was based on motivations and emotions to achieve behavior selection. Controlled variables activate motivations, such as curiosity or fatigue, driven by internal needs or drives when their values goes out the viability range. The motivation with the highest value tries to execute behaviors that can best

contribute to satisfy the most urgent need(s). Thus, the execution of a behavior affects both the external world and the agent's physiology.

With respect to social interaction, Breazeal designed a behavior system for the robot Kismet [60]. This system, represented in Figure 2.4, was developed to be able to support the kinds of behaviors that infants engage in. She defended that, in general, an animal can only pursue one behavior at a time. Each behavior is viewed as a self-interested goal-directed entity that competes against other behaviors for control of the agent. Moreover, each behavior determines its own degree of relevance by taking into account the agent's internal motivational state and its perceived environment.

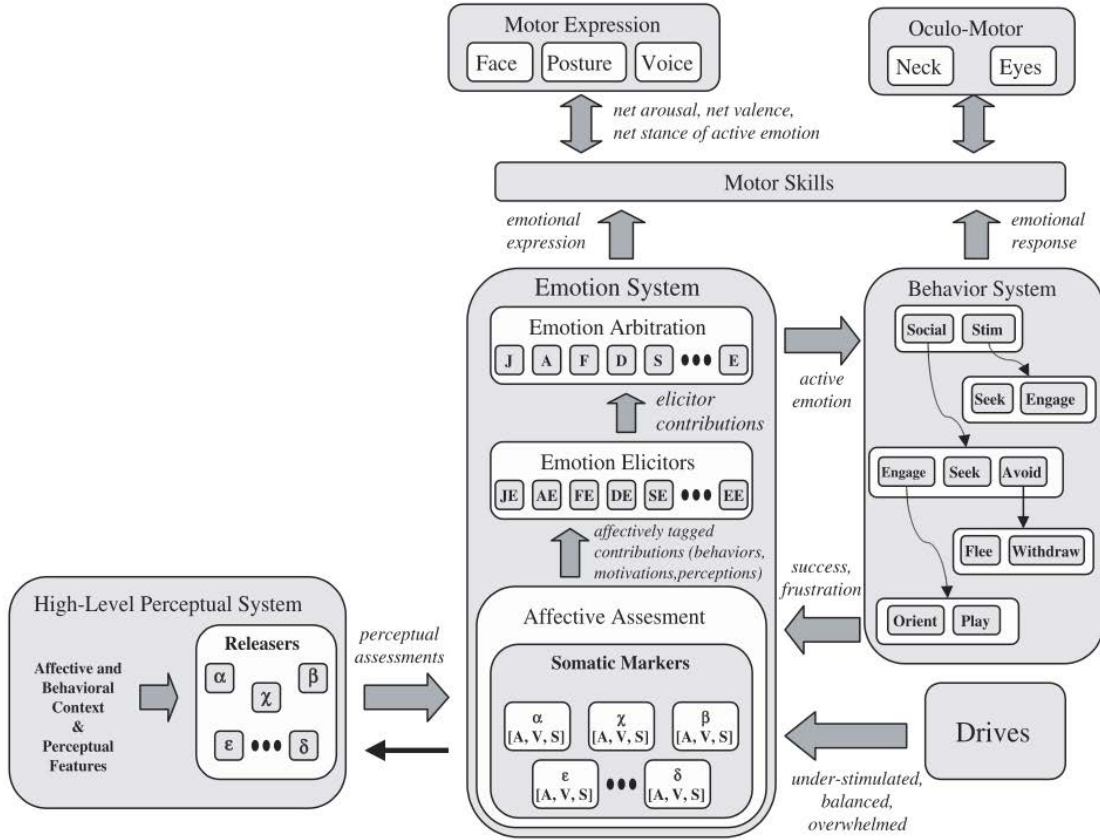


Fig. 2.4 An overview of the emotion system for the Kismet robot.

One more relevant work is the proposed by Vouloutsis et al. [61]. They proposed an Experimental Functional Android Assistant (EFAA) that contains intrinsic needs to socially engage, action repertoire that supports communication and interaction, and the core ingredients of social competencies: actions, goals, and drives. They define as drives the intrinsic needs of the robot. Goals as the functional ontology of the robot,

and depend on the drives whereas actions are generated to satisfy goals. The EFAA agent is made to perform different actions in order to satisfy its drives. They consider that such behavior is adaptive since it allows the system to achieve specific goals, like the satisfactions of a specific drive in a dynamic environment. However, the system they implemented set the ground for a more thorough implementation of behavior selection where the agent can learn to pick the optimal behavior. Figure 2.5 shows an overview of the system from these authors.

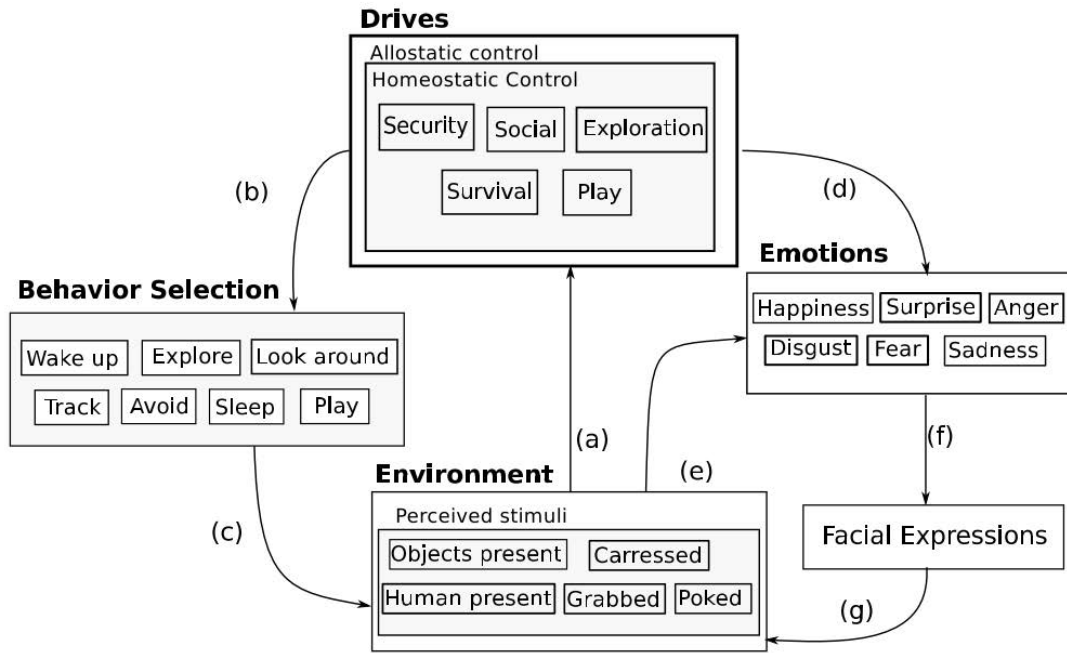


Fig. 2.5 Overview of the parts involved at behavioral level of EFAA.

Current works such made by Cao et al. [62], demonstrated that *homeostatic* systems has evolved to be adapted to the human-robot interaction field. They proposed a hybrid concept for the behavior decision-making process, which combines the hierarchical approach to create an architecture for the NAO robot. The social behavior controller (Fig. 2.6) was developed underpinned by a psychological perspective. Where behaviors are selected based on the external stimuli and the process of maintaining the needs within acceptable ranges. The working of the architecture starts with the *Perceptual system* which interprets data from sensors into abstract stimuli. These stimuli are inputs for the *Homeostasis system*, which manages the needs. Thus, the *Behavior system* generates behaviors by the reactive and deliberative layers and calculates the

affective state. This affective state is used later by the *Emotion system* to generate robot emotions.

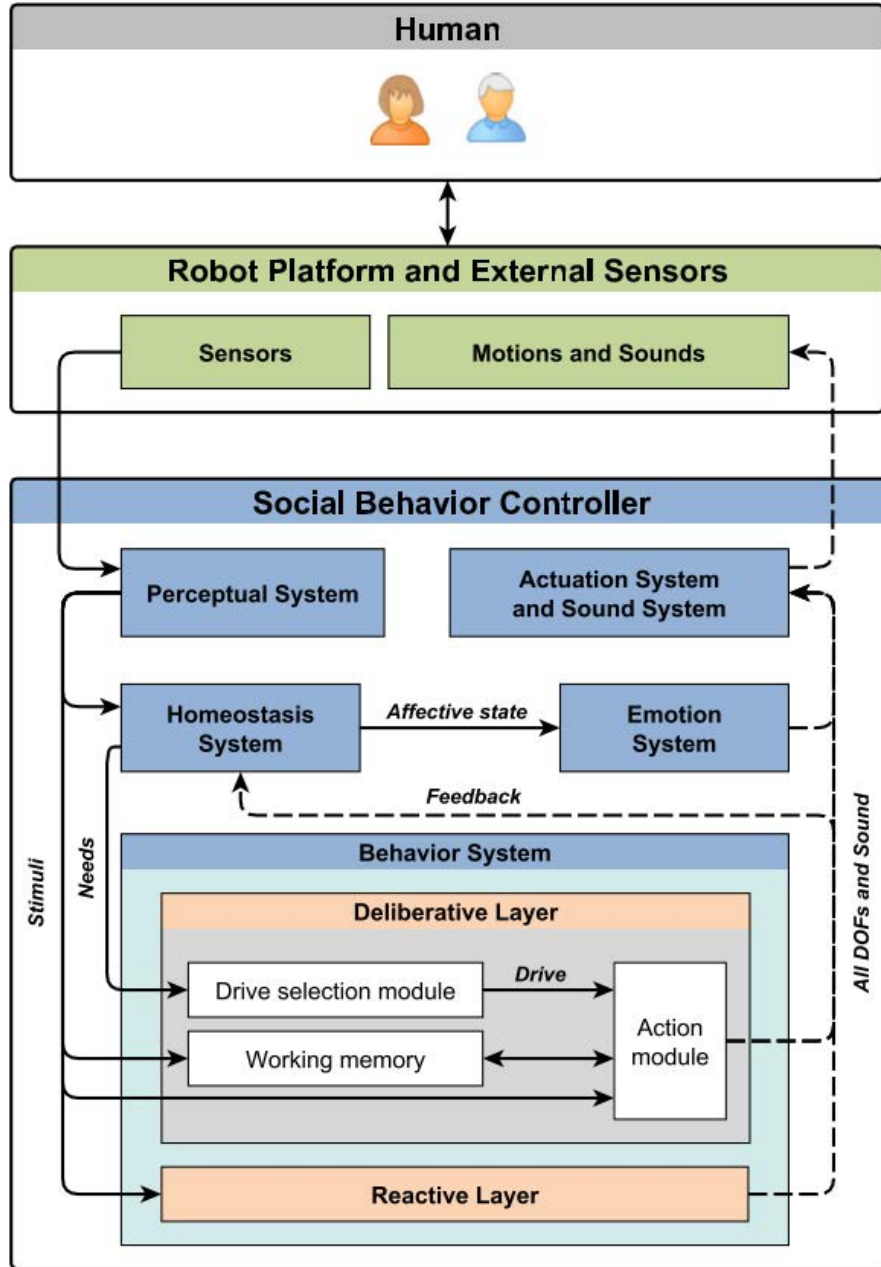


Fig. 2.6 Overview of the social behavior controller architecture.

Although we can find other studies that address biological foundations in cognitive processes such as decision-making or reasoning [63][64][65][66]. This thesis is focused in the model developed by Malfaz [67] that is described as previous works in Chapter 3.

2.3 Decision-Making for HRI

Human-Robot Interaction (HRI) and Collaboration (HRC) are research fields aimed at facing collaborative challenges between robots and humans in complex environments [7]. Robots have been introduced in these environments, such as hospitals, hotels, schools, etc. to help or improve the quality of life.

Traditionally, robots involved in HRI have been essentially based on a master-slave control where the human teleoperates or programs the robot in advance by the technique called Wizard of Oz (WoZ). Moreover, to avoid potential safety problems, industries were the common workspaces where humans usually were not able to interact with the robots. For instance, in a automobile production line where robots execute assembly tasks.

For these reasons, robots have been moving out of laboratory and manufacturing environments to enter into more familiar conditions. In the last few years, a number of social robotic platforms have been developed with the capability of exhibiting social behaviors and collaborating with non-expert users in diverse environments e.g. homes [68], schools [69][70][71][72], office [73][74], hospitals [75][76], museums [77][78][79], and others [80][81][82].

Unlike the traditional robots in environments which are operated by experts and used as tools in industries, social robots are expected to behave in a more natural manner. For example, following we describe some relevant works in this field where behavior selection has an important role in order to improve the autonomy of the robots.

First example in which we can find robots collaborating with people is in *Edutainment* environments. According to Chernova [83], the ability for robots to engage in interactive behavior with a broad range of people is critical for future development of social robotic applications. They showed an example of how the use of games facilitates human-robot interaction research in order to create robust and diverse interaction behaviors. However, we can find robots just for entertainment as a gaming platforms [84]. These robots have the task of playing with their human partners to accompany and satisfy their desire to entertain. On the other hand, robots can be used only for educational purposes. This is the case of the study made by Ramachandran et al. [85], where a social humanoid robot is used to tutor children in one-on-one interactions. They outline an architecture in which the robot uses reinforcement learning to adapt the difficulty of its exercises. The robot, therefore, should be able to detect and recognize the actions and intentions of a student, producing appropriate actions.

Another example is in *Medical* environments, where robots usually collaborates with the personnel involved. In this way, robots can be used in therapy with children with autism spectrum disorders to encourage social interactions [86]. Additionally, the use of therapeutic robots has already seen success using social robots to facilitate social behaviors [87]. Even so, according to Cao [88], despite the numerous robot architectures being developed in this field, very few offer reutilization opportunities in other contexts.

Nevertheless, many social robots in HRI are still operated under the WoZ techniques in which the experimenters spend a significant amount of time to manually control the robot's actions [89][90][91].

The personality of the robot is an arduous task that has to be carefully handled. Therefore, we can find some studies as made by Breazeal [92] related to this topic, in which social robots designers can model the robot to balance the needs of people.

To approach such problem, researchers in cognitive robotics have developed and implemented in some cases, a number of architectures to enable social robots to make decisions more autonomously. This thesis has based the architecture implemented in the model developed by Barber [93] and Rivas [94] that is described as previous works in Chapter 4.

2.4 Conclusion

We presented the relevant works related to bio-inspired decision-making in robotics. Showing different models developed depending on the type of robots applied. We also showed the essential fundamentals about biologically inspired decision-making, describing the relevant concepts applied in this topic.

Finally, we introduced other relevant concepts about the current state for action selection in human-robot interaction. Showing the traditional systems applied in this field and showing the importance of an action selection system for SIR.

Chapter 3

Autonomous Decision-Making

He that has a choice has trouble.

Dutch Proverb

3.1 Introduction

This chapter presents the theoretical principles applied for a biologically inspired autonomous decision-making system designed for robotics architectures and focused on human-robot interaction. Following, the decision-making process is explained in detail. First, we present the general scheme followed as a global view of the process. Later, the states of the robot are described in order to understand how the actions are selected. Then, we introduce the concept of *skill* followed to the learning methodology used. Finally, some conclusions of this work are presented.

3.2 Previous works

The DMS designed for this thesis is a continuation of the previous work presented by Malfaz [95] and Castro [96]. They created a bio-inspired decision-making system based on motivations, where no specific goals are given in advance. Thus, Malfaz designed a DMS (Figure 3.1) based on theoretical principles about emotions and self-learning that was tested on social virtual agents. Later, Castro implemented these concepts in a social robot.

At this point, the relevant biological concepts are going to be presented in addition to the mathematical model of every of them. In this way, the reader is able to understand the concepts along with its representation.

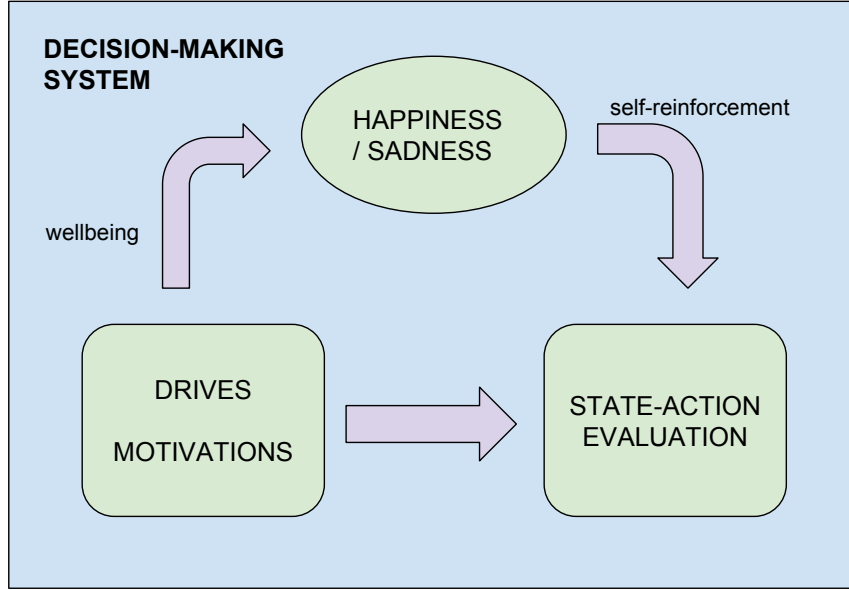


Fig. 3.1 Biological inspired decision-making system developed by Malfaz.

Do not forget that the purpose of our DMS is to have an autonomous robot that has certain needs and motivations. The objective of the robot will always be to maintain all of them as satisfied as possible. For that reason, a learning process is done to select the right action in order to maximize the wellbeing of the robot. This **wellbeing** is defined as a function of its drives and it measures the degree of satisfaction of its internal needs. The wellbeing is modeled as shown in Equation 3.1.

$$Wb = Wb_{ideal} - \sum_i \alpha_i \times D_i \quad (3.1)$$

where α_i are the ponder factors that weight the importance of each drive on the wellbeing of the robot. Wb_{ideal} is the ideal value of the wellbeing which corresponds to the value of 100. Where the values of the needs of the robot (drives) increase, its wellbeing decreases. Thus, drives are inversely proportional to the wellbeing: the lower the drives are, the higher the wellbeing is. The positive and negative variation of the wellbeing is interpreted by the robot as *happiness* or *sadness*.

Thus, **drives** indicates a deficiency or a demand that causes the desire to satisfy this demand or to overcome the deficiency. In this way, drives are often viewed as *homeostatic processes* that motivate actions in order to reach and keep a certain balance. The term homeostasis refers to a state of psychological equilibrium obtained when

tension or a drive has been reduced or eliminated. For this approach, drives are considered as the needs of the robot where the ideal value is **zero** (satisfied). Drives evolve automatically increasing their values by a defined equation until it is reduced by an *exogenous action* or completely satisfied.

The robot has also **motivations** that are those internal factors, rather than external ones, that urge the organism to take action. For instance, a motivation for the robot would be the need to be more social.

Based on this definition, the intensities of the motivations are modeled as a function of its drives and some external stimuli. **External stimuli** are perceptions coming from the environment that alter the tendency to act, that is, the motivations to behave in one way or another. External stimuli for the robot would be for instance a change in the environment such as a user leaves the room.

Depending on the the level of a drive, an intensive or weak stimulus is needed to affect the behavior of the robot. The intensities of the motivations are calculated as shown in Equation 3.2.

$$\begin{aligned} \text{If } D_i < L_d \text{ then } M_i &= 0 \\ \text{If } D_i \geq L_d \text{ then } M_i &= D_i + w_i \end{aligned} \tag{3.2}$$

where M_i is a particular motivation, D_i is the related drive, w_i corresponds to the related external stimuli, and L_d is called the activation level. Motivations whose drives are below their respective activation levels will not be able to lead the robot's behavior.

Motivations are competing continuously among themselves for being the dominant one. The motivation with the highest value is considered the **dominant motivation**. This motivation will determine the internal state of the robot. An important consideration is that a drive below its activation level will not compete for being the dominant motivation.

The global state of the robot will be a combination of the internal and external state. The internal state is determined by the dominant motivation as we explained. But the external state is defined by the relation to every object in the environment.

Every thing in the environment of the robot can be considered as an **object**. Based on this formulation, people would be considered also as an object in the environment for the robot. Objects may be in a concrete state. These **states** are defined a priori because it will be extremely complex to model the world without that previous knowledge. Some examples of states related to a person are: *present* or *absent*. The state of all objects in the environment defines the *external state* of the robot.

In that way, the **action** selected at each time will depend on the final state of the robot and the potential repertoire of actions available in that state. Through the learning process, the robot will be learning what is the best action in every situation. Every action has an **effect**. The effect of an action can be positive or negative depending on the result.

Thus, if a robot executes the action of speaking with a user in front of it, and the user leaves the room instead of starting a conversation. The effect of that action could have been a negative interaction. Otherwise, if a robot executes the action of moving from the door place to the charger and the position is achieved, the effect of that action would provoke a positive effect.

Action's effects will produce then a **reward** in the system. The reward value for one action executed in certain state corresponds to the variation for all drives during its execution, that is, the wellbeing variation. For instance, for an action a , the reward is computed according to Equation 3.3.

$$reward_a = \Delta Wb_a = Wb_{aftera} - Wb_{beforea} \quad (3.3)$$

Considering Equations 3.3 and 3.1, the total reward for an action depends on how fast drives change their values during the execution of that action. Moreover, the positive/negative variation of the wellbeing is interpreted as happiness/sadness for the learning process (Section 3.7).

Summarizing, the decision-making flowchart is shown in Figure 3.2. Where the internal and external states are selected along with the reward of the action executed as inputs of the DMS. The decision evaluator will select then the new action to execute. This is a continuous process that will never stop.

3.3 The Decision-Making process

Once the concepts are clear, here we describe the process of making decisions based on the previous works, that will allow any SIR to act in an autonomous manner when interacting with people.

The common steps followed in the process of decision-making are:

1. Firstly, *the state of the robot is obtained*. The state of the robot is the combination of the internal and the external states. The internal state comes from the status of the drives that combined defines the dominant motivation. On the other hand,

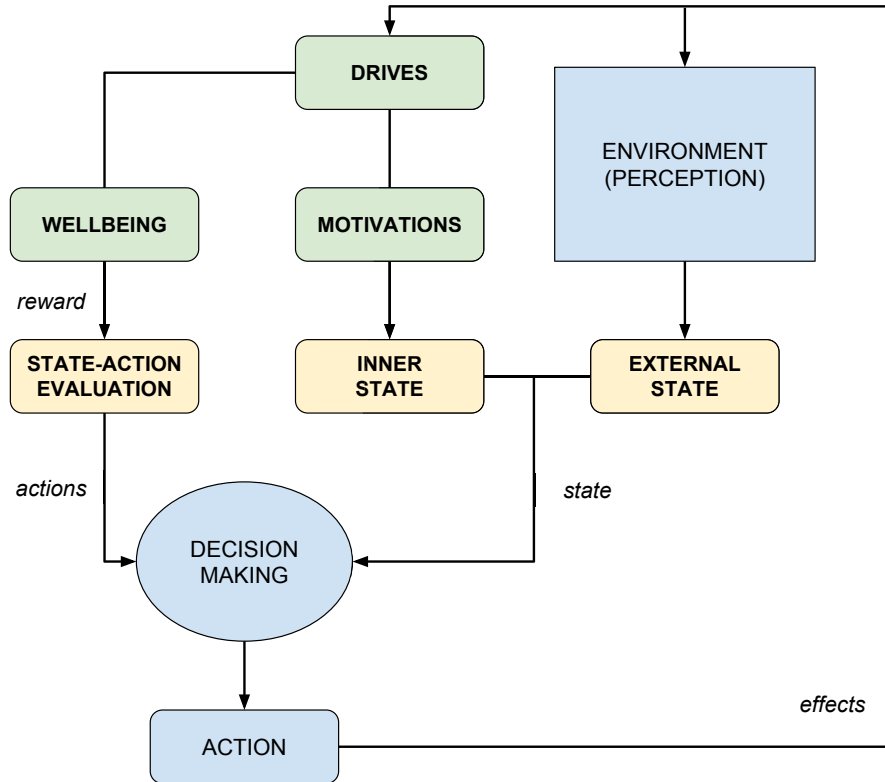


Fig. 3.2 DMS components flowchart.

the external state comes from the status of the person present in the environment. This information is sent to the decision-making evaluator.

2. After that, *the evaluator selects the next action to do*. The process of evaluation will search the available actions for the state of the robot. That means, depending on the internal state of the robot and the external state of the world the robot has limited available actions that can be executed in these circumstances. Every time there is a change in the internal or external states, if there was a previous action executing, the evaluator sends a signal to interrupt the current action. When the action has finished, a confirmation is sent to the DMS. In this moment, a new evaluation is done selecting a new action to execute.
3. Once the action is selected, the action starts executing until *it is completed or interrupted in order to receive an effect*. When the action is selected by the DMS, that action starts doing its functionalities. However, if a signal of interruption is detected by the action will interrupt the execution as soon as possible. Once the

action has finished (correctly or due to an interruption), it will send the resulting effect producing a reward affecting the wellbeing. Moreover, the result of the effect of an action will produce a variation in the drives. Increasing or decreasing their values depending on that result.

4. Finally, *the DMS learns from every action executed*. We use the Q-Learning methodology to learn from interactions. This method uses the last and current information about the scene in which the action was executed and learn from experiences. The knowledge learned will affect to the selection of future actions.

Following, a more extensive description of all these steps are extensively presented in order to understand the DMS.

3.4 The state of the robot

The state of the robot $s \in S$ is the combination of the set of internal and external states (Equation 3.4):

$$S = S_{internal} \times S_{external} \quad (3.4)$$

Where the internal state is defined by the dominant motivation and the external state is related to the user present in the environment. For this thesis, we have limited the number of users in the environment to only one. Reducing in this way the state space and simplifying the problem. Thus, Equation 3.5 shows that the external state is defined by the state of the user that is going to interact with the robot.

$$S_{external} = S_{user} \quad (3.5)$$

For instance, considering the robot has been interacting during a long time and the need of relax is the dominant. If there is a user near to the robot, the global state would be like shown in Equation 3.6.

$$\begin{aligned} S &= S_{internal} \times S_{external} = \\ &= S_{dominantmotivation} \times S_{user} = \\ &= relax \times user(near) \end{aligned} \quad (3.6)$$

Following, we explain in detail how the state of the robot is composed.

3.4.1 External states

The external state is composed by the states of all objects in the environment. However, in this thesis, the robot can only interact with people that is the only “object” for the DMS. Besides, we have focused our research in one by one interactions, so the robot will be able to monitor the state of a specific user.

We have defined seven available states for a user (Figure 3.3). These states have been defined based on the distance of the user regarding the robot. By default, the robot always starts thinking that a known user is in the *absent* state. Afterwards, depending on detectors the state will be changing. We have defined two main methods to transit from one state to another. One depends on perception sensor devices where the information from the world is obtained. And the second one where transitions are modeled as automatic, passing to the next state after a period of time.

In order to control the **exogenous actions**, actions that are made by people but not due to a direct consequence of the robot’s actions. We have modeled the external state of the robot defining the states of a user how this kind of actions. Exogenous actions are responsible of altering the robot and its environment but they are not under the robot’s control. In this way, we control those actions considering the state of a user when approaching or moving away to the robot.

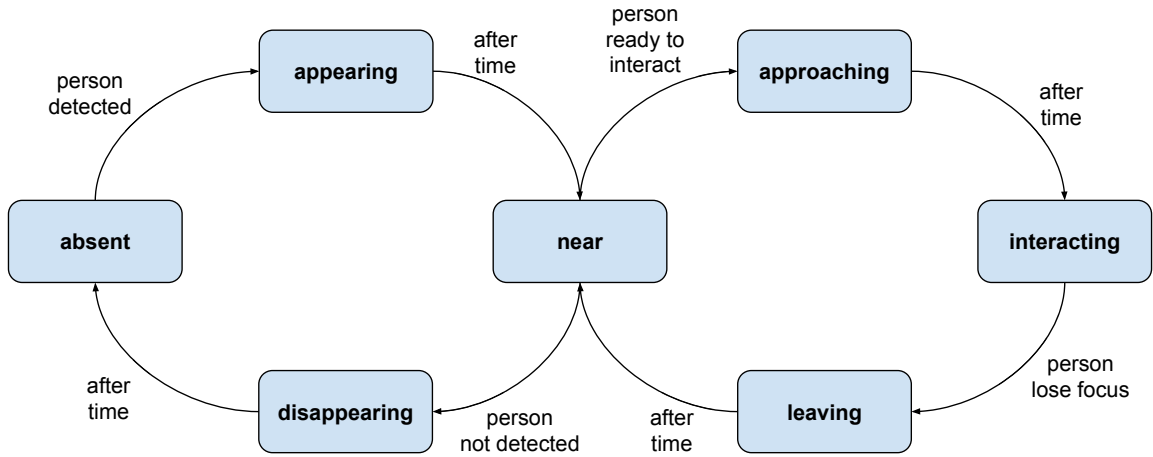


Fig. 3.3 Available states for a user.

Depending on the distance of a user to the robot, the robot is able to control the environment around it. As we have commented, the robot initially thinks that a known user is **absent**. A user is absent when the robot is not able to perceive the proximity

of that user. However, it will start to detect if the user is really absent or present. If the user is detected, the temporal state **appearing** is set to that user, and after a time period defined the user state will change to **near**. The near state is the only that forks in two available states depending if the user moves closer to the robot or goes away.

When the user is ready to interact with the robot, the **approaching** state is set. This state is also a temporal state that after a period of time passes to the **interacting** state. When the user is interacting with the robot but loses the attention or stops interacting, the user changes the state to the **leaving** state - one more time a temporal state.

Alternatively, when the user is in the *near* state, if the robot detects that the user goes away from its environment the user will transit to the **disappearing** state. Another temporal state that produces the user returns to the *absent* state.

It has to be pointed out that the *appearing*, *approaching*, *leaving*, and *disappearing* states follow a policy of automatically transit to the next state after a period of time because, in this way, the robot is able to control exogenous actions from users that may depend or not from an action executed.

3.4.2 Internal state

The internal state is represented by the *dominant motivation* selected (see Section 3.2). The dominant motivation is calculated from all motivations that are competing, those that are above their *activation level*. In Equation 3.2, we described how motivations are formed by two factors: internal needs and external stimuli.

Internal needs, the drives, are fluctuating (increasing or decreasing) depending on their parametrized evolution function. Thus, drives can evolve depending on external signals or predefined parameters. Each motivation is directly connected to a single drive.

Evolution functions are defined by the designer affecting the behavior of the robot. We usually apply mathematical functions (e.g. a linear, or an interpolated function) with a set of predefined parameters to model some drives as well as objects states to determine if a drive has to increase or decrease its value. Drives evolution is determined then by three factors: the *satisfaction time*, the *evolution function*, and the *saturation level*.

In order to normalize the limit values for a drive, we have defined the minimum (0) and the maximum (100) value a drive is able to reach. However, every drive could have a different maximum value that it will be able to reach. That means that each drive can have different saturation levels. Once a drive has reached its saturation level,

it does not exceed this value and remains at it. This approach is made in order to create an emergency mechanism in case that several drives are saturated working as predefined priorities to determine the dominant motivation.

Figure 3.4 shows some examples for drives where it is represented the main parameters. The **satisfaction time** is present at the beginning of the chart. After a drive is satisfied, it does not immediately start evolving, there is a satisfaction time before it is able to increase again. Likewise, the **evolution profile** for each drive may be different. We follow different approximations such as mathematical equations to model the behavior of every drive. Besides, the evolution profile could depend on an external state or even to an event due to an action execution. Finally, the **saturation level** can be found at the end of the chart in which there are different thresholds in order to determine the priorities.

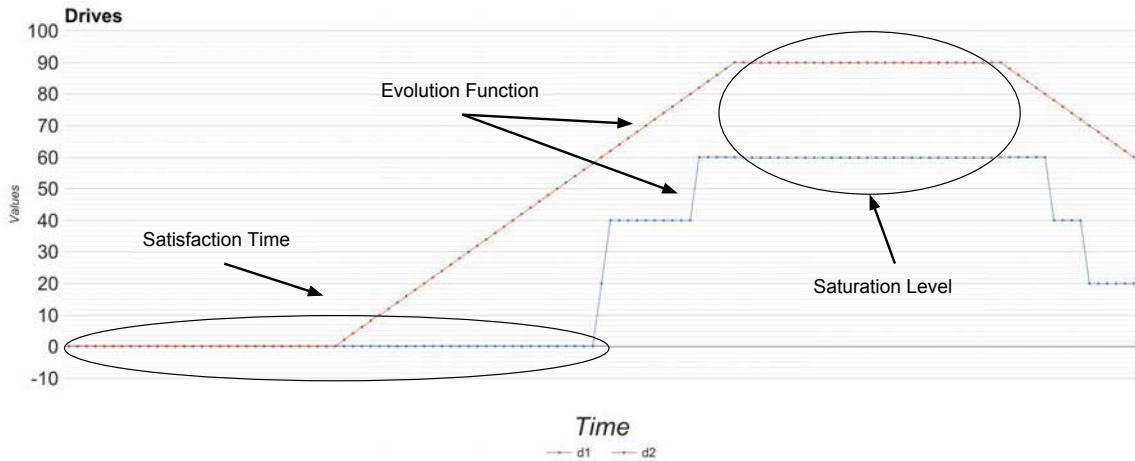


Fig. 3.4 An example for several different drives.

3.5 Selecting the action

The most important step in the decision making process is the action selection. This is critical indeed because an incorrect decision will take a wrong effect affecting to the wellbeing of the robot.

The objective for the DMS evaluator is to correctly select the action that provokes a higher wellbeing for the robot. Remembering how the DMS works (Section 3.3), for the first time the evaluator will be launched to select a random action depending on the state of the robot. After that, it will be continuously waiting for some signal related to

a change on either the internal or the external state. Every time a change is received the evaluator will obtain the best action in the current conditions. Additionally, every time an action has finished, the evaluator will also be launched to select the next action if so.

To evaluate which action is the best, depending on all parameters - the dominant motivation, the states or the user, and the knowledge learned from past experiences, every available action have a probability to be executed. This probability is determined by using the Boltzmann distribution [97]. The probabilities of selecting an action a in a given state s will be determined by Equation 3.7.

$$P_s(a) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum_{b \in A} e^{\frac{Q(s,b)}{T}}} \quad (3.7)$$

where $Q(s, a)$ represents the pairs formed by the state s of the robot and the action selected a that have an associated value which represents the utility of that action in that state for the robot. A represents the set of all possible actions, and T is the *temperature*.

Although we explain in detail in Section 3.7 the learning process followed. At this point, it is necessary to know the concept of *temperature*. Depending on the value of this parameter: a higher value will give the same likelihood of selection to all possible actions, otherwise, a lower value will enforce actions with better value learned. In this way, we are able to adapt the exploration and exploitation level. The value of the temperature is set according to Equation 3.8.

$$T = \delta * \bar{Q} \quad (3.8)$$

where a high value of δ (e.g. 100) implies a high temperature and a low value (e.g. 0.1) will affect the temperature analogously. On the other hand, \bar{Q} is the mean value of all values learned for every action available.

Depending on the final value calculated, if exploration dominates then all actions have approximately the same probability of being selected. Thus, action selection is totally random causing a good exploration phase. On the other hand, if the exploitation predominates then the action learned from previous experiences and with the higher value of reward will be chosen.

Finally, every time the evaluator selects a new action to be executed checking if there was a previous action executing. In that case, the evaluator sends an interruption signal to the active action. Until it is not received the ending signal from the previous action that was executing, the DMS will not execute the new action selected.

3.6 Acting in the world

Once the action is selected, the execution process starts. One of the contributions of this thesis is the continuous evaluation of the best action to execute and, therefore, the need of interrupting the current action. We have redesign the concept of skill adding a more complex definition in order to detect when an action is executing or has to be interrupted.

When an action is selected, the skill associated to the action needs to be activated before sending the goal, if any. Once the skill is started and the goal sent, this begins the action execution. Until the action does not finish, the DMS will be waiting for or deciding if any other action has to be executed. In that case, an interruption signal will be sent in order to finish the current action and to start the new one. If no interruption signal is received, the action will follow its normal course until it finishes.

As we commented, every action has an effect as a reward. Based on this principle, we can check two possible repercussions. An action may provoke either a change in the external states of the world, or cause directly some effects over the drives - the internal state. In this way, the reward will affect directly to the next action selected. The actions that have produced a high reward will be the most probable to be selected after the learning process.

Following, we center in the redefinition of the concept of skill previously presented in Section 4.2, and how the interruptibility has been included. Later, we present the inclusion of memories in our system in order to facilitate the interaction in the DMS.

3.6.1 Skills

We designed skills to have two states: **stopped** and **running**; and two working modes: **continuous** and **conditional**. Figure 3.5 shows the flowchart for both kind of skills and its states.

Starting for the working modes, we refer to a *continuous* skill when the main activity it develops cannot be *stopped* by itself. When a continuous skill is launched, usually at the beginning of the life of the robot, the skill starts automatically to act with no intervention of any external components. This kind of skills can modify its behavior depending on either internal variations or external states but never by the decision-making system.

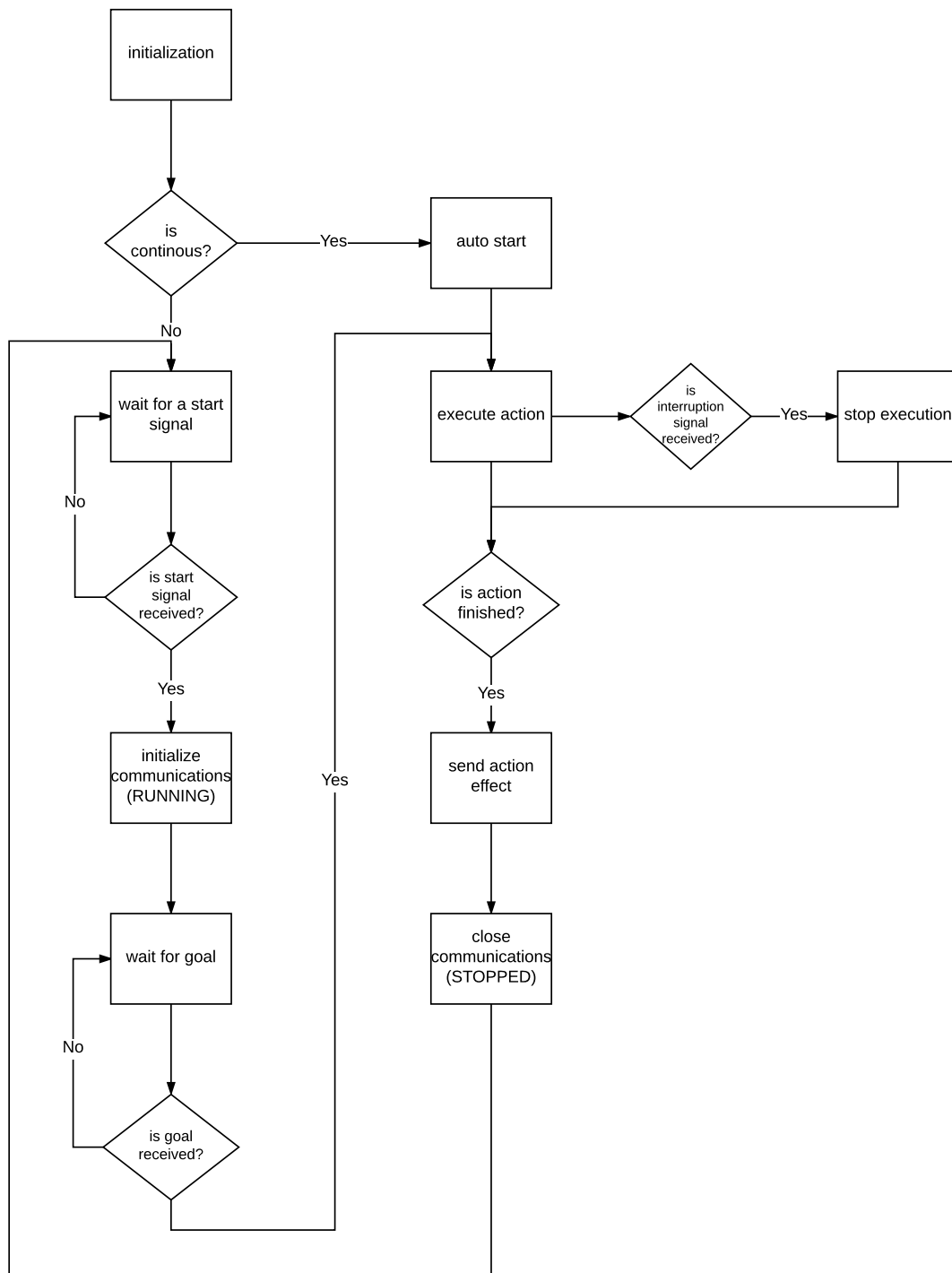


Fig. 3.5 Skills flowchart.

In this way, a *continuous skill* could be some simulated biological behavior in the robot such as the action of breathing or eye blinking. Natural behaviors that humans

do unconsciously and cannot be stopped normally. We could simulate the action of breathing by a simple movement of the body or in a lighted heart by blinking a led. On the other hand, the action of eye blinking could be done by using a screen with some image transitions or mechanical eyes with some kind of actuator.

Alternatively, we refer to a *conditional skill* when it has to receive a goal (or some kind of input parameter) to execute the action. The goal indicates the signal to start acting, being an input parameter such as a position in the world or simply nothing (empty). During the execution of the action, we need to define the breakpoints, if any, that would manage the interruption signals. Once the action has finished, in a natural or interrupted way, it always will have a consequence either positive or negative.

We have defined a policy of action effects based on the state of the skill and the results of the actions. If a robot is in the state *stopped*, we have defined that the consequence will be an **error** because it is not possible to execute while the skill is not *running*. On the other hand, while the skill is *running*, if a goal is received the action will start executing with two possible results depending on the completion of the skill: **success** or **fail**.

Therefore, an example of *conditional skill* would be that one that controls the navigation of the robot. That is, when a user says for instance “Go to the door”, a new goal is established and the robot has to analyze it executing the action. If the skill was *stopped*, the action could not be performed so the consequence is an *error*. Nevertheless, if it is in the *running* state, the consequence would be conditioned depending on the completion of the skill. Thus, if the robot is able to access to the given position in the room, the result of the action will be a *success*. On the other hand, if it is impossible to achieve that goal, it will result in a failure (*fail*).

Additionally, we have included the possibility of interrupting the conditional skills at any time. This improvement offers to the DMS much more modularity in addition to an action selection in real-time HRI.

The concept of interruptibility can be better understood by an example. Imagine that a robot has to control its mobile base, calculating the path from one point to another and avoiding obstacles. But, imagine now that a known person moves closer to the robot in an intermediate point of the path. One case could be that the person would greet continuing later with his own tasks. However, another case could be that the person is a good friend and he would like to start a conversation, aborting completely what he was previously doing. In both cases, the common factor would be that there is an interruption in the active action they were doing.

For situations like this, we provide a realistic “interrupting” mechanism where skills can be interrupted at any time. But it is true that actions cannot be interrupted at any point. We refer that it always exists a minimum unbreakable *atom*. Based on this assumption, every action is made by several unstopable atoms. Every atom has to be completed individually but it is possible to interrupt the action between those atoms. In this way, it is really important when implementing a conditional skill to decompose every action in atoms to define the interruptible points where the action would be interrupted in a natural way.

Figure 3.6 shows the concrete process of interrupting an action. Once the goal is received, the action starts executing its normal flow atom by atom. If no interruption signal is received when executing, the action will finish correctly with a successfully consequence. Otherwise, when the interruption signal is received the action waits for the end of the active atom and breaks sending a consequence that the action has failed.

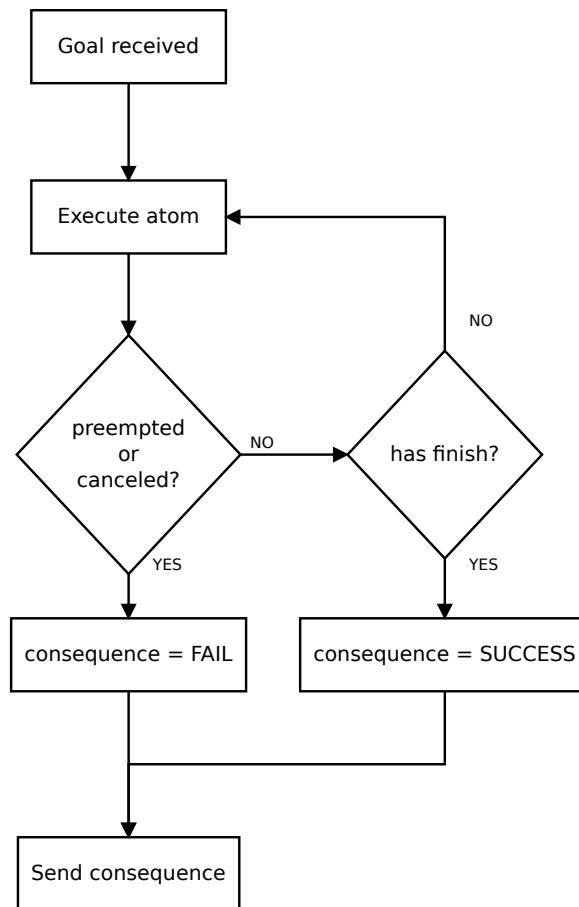


Fig. 3.6 Interruption process flowchart.

Using this solution, our decision-making system will now be able to contain the way the robot-driven interactions can be made being capable to react to unexpected situations.

3.6.2 Memories

Memories are one of the essential components for a cognitive architecture [98]. They are involved in how the information is stored and managed for future use. For instance, during the decision-making process some skills could base its behavior depending whether a known or an unknown user is near to the robot.

Before going into detail of every memory we have included, Table 3.1 from Wood et al. provides a brief but clear definition of terms about memories [99].

Term	Definition
Memory	The capacity to use previous experience to inform subsequent behavior.
Log-term memory (LTM)	Temporally indeterminate, independent of specific task demands.
Short-term memory (STM)	A functionally distinct memory structure, finite in capacity and retention period, bounded by context and task demands.
Working memory (WM)	A variant of the STM concept providing temporary retention of task relevant information.
Procedural memory (PM)	Non-consciously accessible memory e.g skilled motor behavior, habits and stimulus-response conditioning.
Declarative memory (DM)	Consciously accessible information.
Episodic memory (EM)	Personally experienced event information, spatially and temporally organized.
Semantic memory (SM)	Context independent information, facts and concepts.

Table 3.1 Memories definitions.

Memories are well justified in cognitive architectures, but there are recent studies that also justify the necessity in long-term social human-robot interaction [100]. According to Baxter et al., the robot adaptivity relies on the capacity to allow prior experience to influence current and future behavior. In this way, the robot may store relevant information that may be used to learn or to personalize its behaviors depending on the interactions.

Following, we present the relevant types of memories that we have used in this thesis.

Starting from Short-Term Memories, the **Working Memory** (WM) represents a cognitive system with a limited capacity that is responsible for the transient holding, processing, and manipulation of information. Working memory is an important component for reasoning and the guidance of decision making and behavior.

As we have commented, this kind of memory is a STM buffer that allows for the manipulation of stored information, while a common STM is only involved in the short-term storage of information and does not entail the manipulation or organization of material held in memory.

Many theories of working memory exist, in which they agree on several key properties [101]. One of these properties is the *limited capacity* of the working memory system. The estimated capacity about the number of “chunks” that can be stored and used is approximately “seven plus-or-minus two”. A second key property is that contents have to be *readily accessible* to other cognitive processes. This property may be adapted to retain only the information that is most important for influencing behavior in the current situation. Another related key property is the *volatility* of contents. In this way, the content may be updated and manipulated quickly and, of course, removed at any time.

This memory is usually implemented as residual data that is always available while the system is running. In this sense, the working memory is modeled in this thesis as a kind of blackboard where any skill or even the decision-making process could save and read trivial information useful for a better interaction. Examples of information stored here could be the data from a laser device, a camera or even a touch sensor.

On the other hand, we have the **Long-Term Memories**. LTM are a permanent repository of durable knowledge. This knowledge can come from learning, from processing the information stored in STM, or it can be given a priori. In LTM, non-conscious, procedural (or implicit) memory underpinning skilled behavior, habits and conditioning, is differentiated from consciously accessible, declarative (or explicit) memory for facts and information [99].

Based on the work by Wood et al. [99], three basic properties of memory function may be identified:

1. Memory is *fundamentally associative*. This implies that there is no requirement for a global ontology or semantic information.
2. The memory system *does not require the explicit storage of semantic information*. Associations in this context may be regarded as multi-modal constructs that are not meaningful in themselves.
3. The memory system is a *functionally distributed* system. Again negating the necessity for a global ontology into which all modality-specific information needs to be encoded.

For this thesis, we have focused on two subtypes of LTM: **procedural** and **episodic**. We find that it is not strictly necessary a semantic memory, and therefore we have not found the need to include it in our work. Next, the explanation for the memories implemented is presented.

The **Procedural Memory** (PM) is a part of the long-term memory that concerns the storage of non-consciously accessible information. This is responsible for knowing how to do things, also known as motor skills. As the name implies, procedural memory stores information on how to perform certain procedures, such as dancing, talking or waiting.

According to Wood et al. [99], non-declarative procedural memory is thus subdivided into four types: skills and habits, priming, classical conditioning and non-associative learning.

In this thesis, this memory will be considered as preprogrammed skills available to execute at any time. For instance, the ability to wait doing nothing could be an example of this kind of memory.

Additionally, the **Episodic Memory** (EM) refers to a person's (or a robot in our case) unique memory of a specific event, so it will be different from someone else's recollection of the same experience. This memory can be viewed as a store of temporally organized events preserving their temporal-spatial relations. By this definition, it is usually said that EM consists of the recollection of linked *what*, *where* and *when* information.

As Deutsch et al. said, "*knowledge about already experienced situations improves the quality of decision making*" [102]. For that reason, we believe that in social HRI it is really important to store and be able to learn about previous experiences.

For the purpose of this thesis, the episodic memory have been included as modular databases that store information about past interactions with people and skills are able to manipulate, update or remove. Thus, the last time and place the robot had a conversation with a specific user is an example of the data stored in this kind of memory.

Until now, as we have stated before, there is no need to include the **Semantic Memory** (SM) in the decision-making process due to the current requirements.

Chapter 4 explains in detail how we have included and implemented these memories in the robotic architecture.

3.7 Learning how to act

In general, learning is important because it provides us the capacity to adapt to new situations. There are many determinant factors that can lead us to make bad decisions. However, when we or robots do an action for the first time, we do not think a priori if the action has a risk or whatever kind of future consequences.

One of the aims of the DMS is to provide the robot with a mechanism for learning how to behave in order to maintain its needs within an acceptable range. That means to keep the wellbeing of the robot as higher as possible.

For this purpose, we select **Reinforcement Learning** (RL) as the supervised learning technique to include in the DMS. RL is an area of *machine learning* inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. The goal of RL is then to maximize the total expected reward. RL differs from standard *supervised learning* in that correct input/output pairs are never presented.

The decision-making scenario for the reinforcement learning framework (Fig. 3.7) starts with an agent that interacts with its environment in discrete time steps. At each time t , the agent that is in a state s_t chooses an action a_t from the set of actions available A , which typically includes a reward r_t . The environment moves to a new state s_{t+1} and the reward r_{t+1} associated with the transition (s_t, a_t, s_{t+1}) is determined. The goal of a reinforcement learning agent is to collect as much reward as possible. The agent can choose any action as a function of the history and it can even randomize its action selection.

However, the learning process included in the DMS is the **Q-Learning** algorithm. This algorithm consists in an agent that is in a **state** and executes an **action**. In that moment, there is a transition to a **new state** receiving a **reward**. The state will

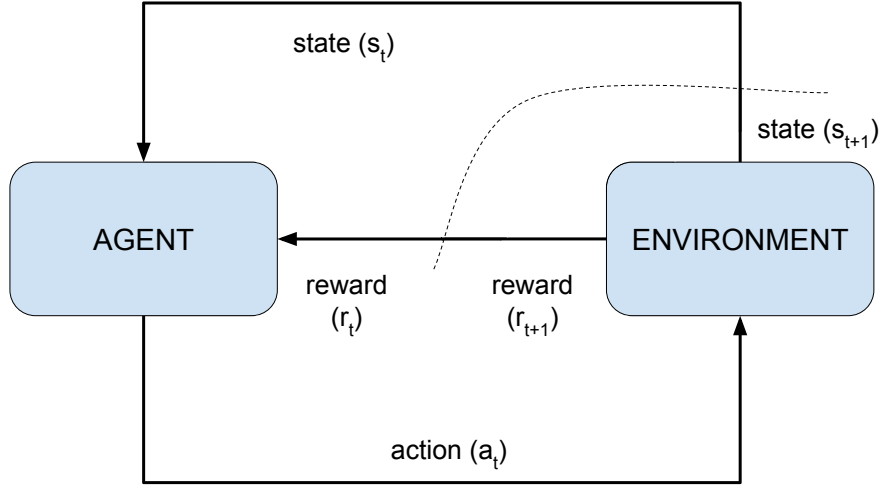


Fig. 3.7 Reinforcement Learning framework.

be determined in relation to the internal and external states, that is, the dominant motivation and the users. Potential actions are restricted by that state. Thus, the learning algorithm will update the **Q-values** for that combination of parameters after an action is executed, being recalculated according to the reward obtained from the previous and the new state. These *Q-values* represent how good a particular action will be at a particular state. Following we will look at each step of the process in a bit more detail.

As we explained, the goal of this algorithm is to calculate the Q-Values for every state-action pair. Q-Values are defined as the expected reward for executing an action a in the state s . Every $Q(s, a)$ is updated according to Equation 3.9.

$$Q^{user_i}(s, a) = (1 - \alpha) * Q^{user_i}(s, a) + \alpha * (r + \delta * V^{user_i}(s')) \quad (3.9)$$

Where $V^{user_i}(s')$ is calculated as shown in Equation 3.10.

$$V^{user_i}(s') = \max_{a \in A_{user_i}} (Q^{user_i}(s', a)) \quad (3.10)$$

The super-index $user_i$ indicates that the learning process is made for the user i . Thus, $s \in S_i$ is the global state of the robot in relation to the user i and $s' \in S_i$ is the new state. A_{user_i} is the set of actions related to that user. Parameters r , δ and α

Autonomous Decision-Making

are respectively the reward or reinforcement received from executing the action, the *discount factor* and the *learning rate*.

The discount factor δ defines how much expected future rewards affect decision now. A high value of this parameter gives more importance to future rewards. On the contrary, a low value gives much more importance to current reward.

On the other hand, The learning rate α controls the weight provided to the reward from the action made. This parameter gives more or less importance to the learned Q-Values than new experiences. A low value implies that the robot is more conservative and therefore gives more importance to past experiences. On the contrary, a high value makes that the agent values the most recent experience.

In order to help the robot learning, the Q-Learning offers two complete procedures where the robot can start exploring available actions many times. And later, it can exploit the acquired knowledge to make better decisions. This is called the *exploration vs. exploitation dilemma* previously mentioned in Section 3.5. We apply this condition to help the robot to be more social, learning from experience. For that reason, the balance between exploration and exploitation will depend on the *temperature* factor and the learning rate (α). Initially, the learning rate will be set to the higher value, which means the most recent data are quite relevant during exploration. After that, at some point it will start decreasing until its minimum value, that means the exploitation phase has started and the Q-Values will not change anymore.

The Q-Learning algorithm updates the Q-Values after an action is executed. Every time a Q-Value is updated, it is referred as an iteration. The pseudo-code for the algorithm is detailed in Algorithm 3.1.

```
1 procedure compute_object_qlearning()  
2   initialize all Q-Values to 1  
3  
4   repeat for each iteration  
5     require:  $s \leftarrow$  current state  
6     require:  $a \leftarrow$  executed action  
7     require:  $user_i \leftarrow$  present user the action is executed with  
8     require:  $s' \leftarrow$  new state  
9     require:  $r \leftarrow$  reward  
10  
11     collateral_effects  $\leftarrow$  0  
12  
13     for all  $user_j$  do  
14       # collateral effects do not consider the user  
15       # that the action was executed with  
16       if  $user_j \neq user_i$  then  
17         max_q_state  $\leftarrow \max[Q^{user_j}(s_{user_j}, a)]$   
18         max_q_new_state  $\leftarrow \max[Q^{user_j}(s'_{user_j}, a)]$ 
```

```

19      collateral_effects  $\leftarrow$  collateral_effects + (max_q_new_state -
    max_q_state)
20      end
21      end
22
23      value_user_i_new_state  $\leftarrow$   $\max[Q(s'_{user_i}, a)] + collateral\_effects$ 
24
25      q  $\leftarrow$   $Q(s_{user_i}, a_{user_i})$ 
26      new_q  $\leftarrow$   $(1 - \alpha) * q + \alpha * (r + \delta * value\_user\_i\_new\_state)$ 
27
28       $Q(s_{user_i}, a_{user_i}) \leftarrow new\_q$ 
29      until learning ends
30  end
31

```

Algorithm 3.1: Q-Learning algorithm in pseudo-code.

3.8 Conclusion

We presented the previous works in which are based the decision-making process developed in this thesis. The main contributions are stated in this chapter where the changes associated to the internal and external states are described. Including the adapted concept of skill where memories provide a mechanism to give more personality to HRI skills. And emphasizing the advantage of the use of interruptions when actions are executing. That makes the robot able to adapt itself to situations when interacting with people. Additionally, the learning methodology is adapted for the HRI purpose that includes a state reduction in order to simplify the process.

This design has been developed in order to maximize the interaction in a social manner. Applying bio-inspired methodologies that makes the DMS a complete system for robotic architectures.

Chapter 4

The Robotic Architecture

Those who can imagine anything,
can create the impossible.

Alan Turing

4.1 Introduction

This chapter presents the implementation of the proposed DMS and the robotic architecture. In addition to the explanation of every component in the architecture, our focus is in the implementation of the concepts explained in Section 3 about the DMS. We also present the previous works in which we are based our implementation.

For researchers involved in the field of robotics, there is no doubt that creating a robot, from hardware to software, is one of the most challenging works nowadays in addition to the high-level capabilities such as a DMS. One of the most important decisions then is to decide the components the architecture is going to include. Traditionally, architectures in robotics have been always related to mechanism of control. Nevertheless, software is nowadays very relevant in real-time systems because software abstraction paradigms offer a wide range of new possibilities. Moreover, software engineering is being applied more and more in robotics improving the way developers can program their systems in a standardized way.

This chapter gives to the reader then a global view from low to high-level layers of the system.

4.2 Previous works

Before explaining the implementation of the system, we describe the previous works related to the robotic architecture that uses concepts related to this thesis. The following architecture presented is an example of how the high-level capabilities such as decision-making can be included against traditional planners.

The Automatic-Deliberative (AD) control architecture is a hybrid control architectures that has been defined theoretically [93] and implemented practically [94] being used for more than 10 years in social robots. This architecture was inspired on the ideas of modern psychology expressed by Shiffrin and Schneider, considering two main levels - the automatic and the deliberative [93]. Communication between both levels is bidirectional by events storing common information in Long-Term and Short-Term memories.

Originally (Fig. 4.1), the implementation of the AD architecture was made with the two well differentiated layers: the **automatic** and the **deliberative**. The automatic layer is responsible for the automatic processes, that is, obtaining the information from sensors and controlling the actuators in the low-level. On the other hand, the deliberative layer is responsible for the reflexive processes, that is, planning the sequence of actions to execute in every moment.

Communication between layers is made by memories and events. There are two kinds of memories defined: the **long-term** (LTM) and the **short-term** (STM). The STM is a temporary memory where the important information is stored and share between both layers. This memory allows to register and eliminate data for future use. The current and the previous value of a data structure is stored as well as the date when saved. This is based on the blackboard pattern. On the other hand, LTM is a permanent memory that can be filled with data either from the STM, the learning procedure, or given a priori. In this case, the LTM information is only available for the deliberative layer.

Additionally, **events** are used for synchronizing the information between layers. An event is an asynchronous signal for coordinating processes by being emitted and captured.

Thus, the automatic layer is implemented as skills which are related with sensors and actuators, that is, physical devices. And the deliberative layer is also implemented as skills which are based on the actions the robot can do. In this case, it may be only one deliberative skill active at the same time.

About memories, STM is implemented as several different types of data that can be accessed in any moment. On the contrary, LTM is implemented as a database and files

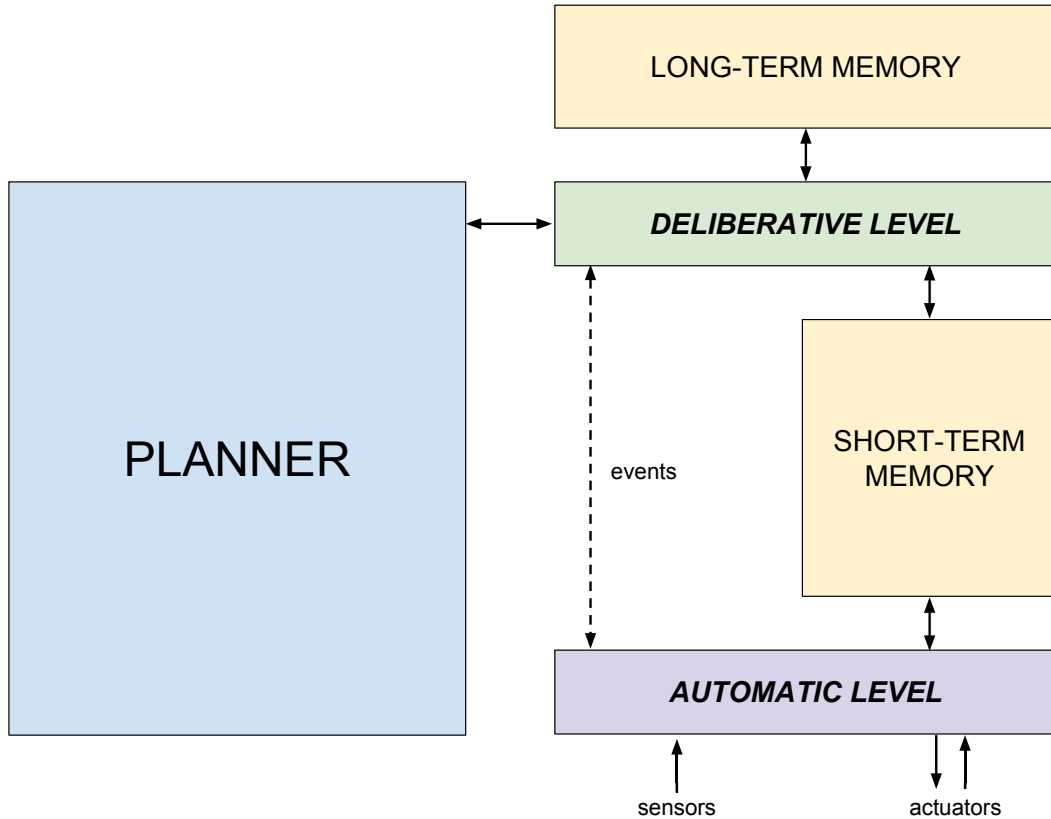


Fig. 4.1 Original AD Control Architecture.

which contain information such as data about the world, the skills, and grammars for the automatic speech recognition module. Lastly, events are emitted with an attached parameter that can be only an integer.

At this point, we can say that one of the cornerstones of this architecture is the concept of **skill** [103] (also called *actions* or *behaviors* in other architectures). A skill describes the global behavior of a robot task or action. This has some special features that should be pointed out:

- It has three states: ready, activated, and blocked.
- It has three working modes: continuous, periodic, and by events.
- Each skill is a process.
- It represents one or more tasks or a combination of several skills.
- It has to be subscribed at least to an event.

Thus, skills are implemented as a class hiding data and processes that can be running in one of the three states and in one of the three modes. Until this moment, skills could only be activated by: other skills or a planner.

Nevertheless, to develop an architecture is a constant work that has to be revised in the time. For that reason, the architecture was upgraded by introducing some biological foundations to provide a high level of abstraction in the action selection [104] (Fig. 4.2) activating skills in a autonomous way. This decision-making system included, previously commented as a work developed by Castro [96], would add more flexibility and new functionalities on how the skills could be executed in the real robot.

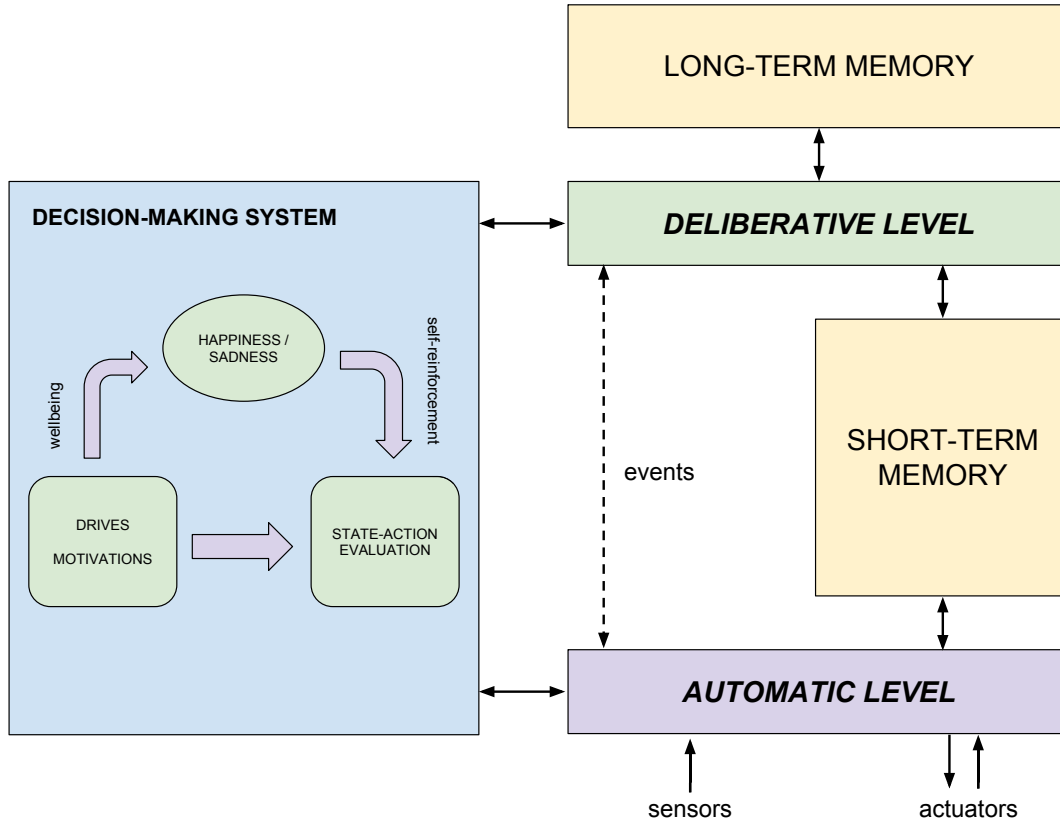


Fig. 4.2 AD Control Architecture including the DMS.

4.3 The Software Architecture

This section introduces the concepts applied to the robotic architecture and its implementation. It also presents the software engineering techniques we have used in

order to add more modularity and portability, such as software *Design Patterns*, *Agile methodologies*, and the *Robot Operating System framework*.

Following these concepts we propose a methodology for developing components in a robotic architecture which aims at offering flexibility, adaptability and a huge grade of modularity when creating high level capabilities for a SIR.

One of the main objectives of this section is to promote reusing software for different robots, specially when they have distinct types of hardware devices. For this purpose, we have arranged the different modules in the software architecture in two main layers: the low- and the high-level (Figure 4.3).

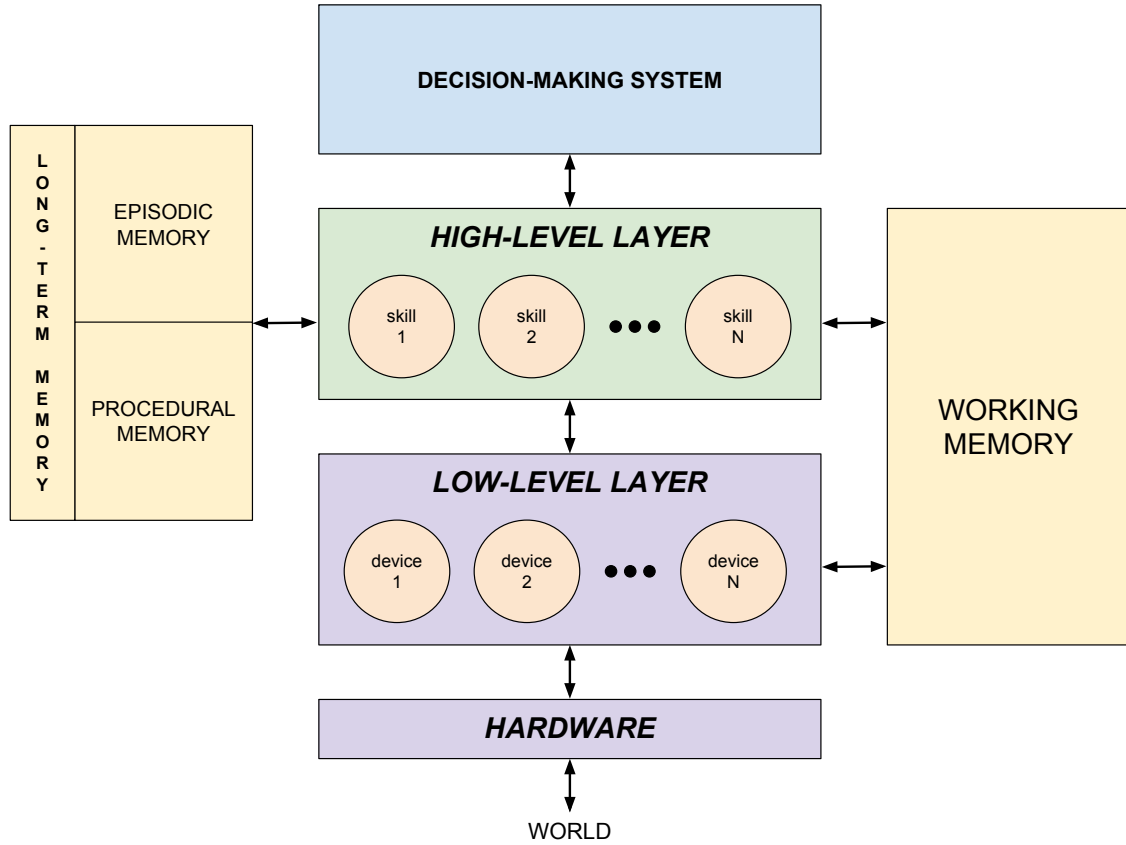


Fig. 4.3 General scheme for the software architecture in layers.

We achieve in this way a multiplatform system when hardware is independent from the low-level layer that is the responsible for communications with the devices. Besides, we offer a standard interface with the rest of the components of the architecture. The low-level will have all specific nodes for controlling those hardware devices included in each robot.

On the other hand, the functionalities developed in the high-level layer will be related with specific actions or behaviors the robot could do. This layer is able to communicate with the LTM depending on the information that is required.

The DMS is decoupled from the high-level layer with the goal of being applied in robots with a different repertoire of skills. Thus, the DMS is controlling and deciding what skill to launch independently of the hardware.

Both layers can use the STM to read and write relevant information that will be used during the life of the robot. That information, depending on the relevance, could be stored in the LTM if the skill deems it necessary.

4.3.1 Software Methodologies Included

The design and development of a software architecture for a robot is a complex task as we commented in last section. For that reason, we present well-known techniques commonly used in software development which can help us to implement a robotic architecture efficiently. Firstly, each technique is presented. Following, we explain its benefits and how to apply them in robotics.

The work presented in this thesis is based on several concepts from software engineering. Object Oriented Programming (OOP) is one of them, but since it is a very well-known paradigm in robotics, we do not focus on it and just use it. Nevertheless, there are other concepts that can benefit the software development in robotics: **Design Patterns** and **Agile Software Development**. These two paradigms are described in the following subsections.

4.3.1.1 Design Patterns

In software engineering, a design pattern is a general reusable solution to a commonly occurring software problem within a given context in the process of software design. *Design Patterns* are commonly applied in the industry of software but they are not used in robotics frequently. Mainly there are two very interesting Design Patterns that can be applied to robotics: the Interface Pattern, Dependency Injection, and the MVC Pattern [105].

Interface Pattern

This pattern belongs to the called *Fundamental Design Patterns* (FDP). FDP are called fundamental because they form the basic building blocks for the rest of patterns. Several of other patterns and modern applications draw on these patterns in one way or another.

The *Interface Pattern* is intended to facilitate the re-usability and abstraction of the software. It describes the type of communication between entities, i.e. the interfaces, and the entities have to handle the exchanged information. An interface defines the declaration for the operations of an entity, it also sets the communication boundary between two entities, in this case two pieces of software. The main idea of an interface is to separate functions from implementations. Interfaces are recommended when designers want to specify how classes exchange messages, when we have to switch the implementation of a module during run-time, or when at design-time we do not yet know which implementation you will use at compile-time.

We apply this pattern when we have robots endowed with the same kind of devices, sensors or actuators, but different models. For instance, consider two robots with different distance sensors each one. Both sensors have the common functionality of providing the distance in meters. If we apply the IP (Figure 4.4), we define a common interface: *DistanceSensorInterface*. In this situation, the low level software controlling the distance sensors have to comply with the definition of the interface. Following this approach, a higher level module that needs to communicate with the distance sensors do not need to know the precise model of sensor, just the interface.

Considering that we could have to change the type of sensor in any moment, or reuse the higher level software that uses the distance sensor data in multiple robots, it is preferable to have an interface with the declaration of the function that can be applied without knowing its implementations.

Dependency Injection

This pattern belongs to the *Architectural Patterns* (AP) group. AP provides reusable models and methods solutions to a commonly problem in a software architecture.

Dependency Injection is based on the “Inversion of Control” (IoC) principle. This is related to the way in which an object (i.e. the instance of a class) obtains the references to its dependencies - the object gets its dependencies through (i) constructor arguments, (ii) through setter methods, or (iii) interface methods. Therefore, there are 3 forms of DI: setter-, constructor- and interface-based injection. DI pattern is recommended to be used in the next situations: when the coupling between components needs to be reduced; when designers want to save time not having to write repetitive factory code over and over again; or when designers are expecting to run controlled unit tests. With dependency injection, testing can begin very early in the development cycle.

In robotics, we can apply DI when we have to control a device but we do not know the specific model, we just know the type of device. For instance, guess that we want to control a motor. We have a class that provides the common functionalities

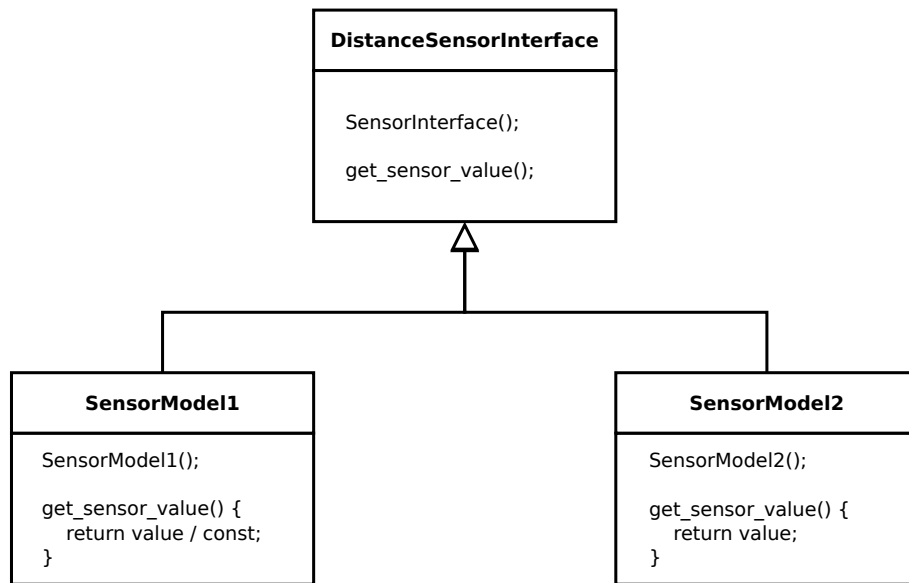


Fig. 4.4 UML Class Diagram applying the Interface Pattern to an example of 2 different distance sensors.

to any motor, for instance like an interface. Moreover, it depends on another class, *MotorDriver*, where the particular primitives are implemented. The *Motor* class could have directly an instance of the motor driver class. Then, we could create the new instance calling directly the sentence to create a new object for the driver as it is shown in Listing 4.1. This approach implies that a change of implementation will require a change on the code.

```

1 class Motor {
2     MotorDriver _md = new MotorDriver();
3 }
    
```

Listing 4.1 Without Dependency Injection.

However, if we consider that we can have many different motor drivers depending on the model, we can define now an interface class (*MotorDriverInterface*) to abstract all of them. When we use DI, as shown in Listing 4.2, we create the instance of a motor driver by its interface, and then pass that instance by the constructor of the class.

```

1 class Motor {
2     MotorDriverInterface _md = null;
3
4     Motor(MotorDriverInterface & md) {
5         _md = md;
6     }
    
```

```
7 }  
8  
9 main() {  
10     MotorDriverInterface mdi1 = new MotorModel1();  
11     Motor m1(mdi);  
12  
13     MotorDriverInterface mdi2 = new MotorModel2();  
14     m1.set_driver(mdi2);  
15 }
```

Listing 4.2 With Dependency Injection.

This is an example of how to use this kind of pattern when you want to reuse code for different devices with a common functionality in robotics.

Model View Controller (MVC) pattern

The *MVC pattern* is based on three types of components in an application, **Models**, **Views**, and **Controllers**. Because these objects are separated by abstract boundaries, community usually say that it is more a paradigm¹ rather than an actual pattern. This pattern also belongs to the Architectural Patterns (AP) group.

Model component hold data and define the logic for manipulating that data. Model objects are not directly displayed. They often are reusable, distributed, persistent and portable to a variety of platforms. *View* objects represent something visible in the user interface, like buttons, images, etc. The *Controller* object acts as a mediator between the Model and Views. A Controller component communicates data back and forth between the Model and the Views. This pattern should be used almost in every GUI² application - of course depending on the application some classes might be coupled tighter than others, however it is generally always a good idea to structure your application according to MVC.

The MVC paradigm can be applied to robotics when a GUI is required. For instance, a teleoperation GUI for controlling the velocity of a motor by hand. The View, as showed in Figure 4.5, would contain a *label* and a *slide* to set from 0 to 100 the velocity. The Controller in this case would set the properties of the GUI, show the GUI to the user, set the communications between the objects of the GUI, and set the communications with the framework which control the motor. In this example the Model is the datum exchanged by the motor and the GUI.

¹A programming paradigm is a fundamental style of computer programming, serving as a way of building the structure and elements of computer programs.

²Graphical User Interface

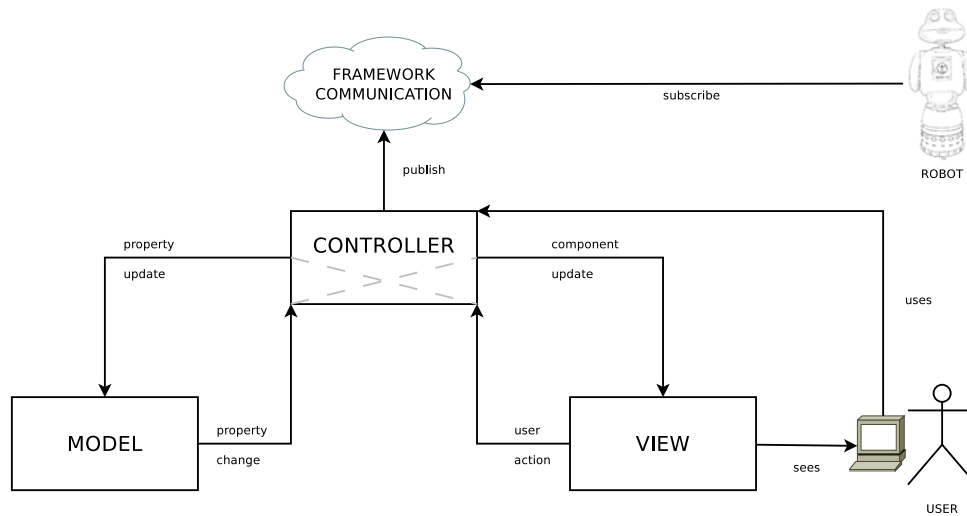


Fig. 4.5 Model-View-Controller Pattern example.

4.3.1.2 Agile Software Development

Agile Software Development (ASD) includes some software development methods in which requirements and solutions evolve through collaboration between cross-functional teams. These methods provide a continuous opportunity to assess the direction and progress during throughout the development life-cycle. *ASD* promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to changes. These are very appealing features in robotics. Consequently, we consider these methods: **Test-driven Development**, **Unit Testing**, and **Mocks**.

Test-Driven Development (TDD)

TDD [106] is an evolutionary approach to software development which combines “Test-First Development” (TFD) and refactoring in order to produce clean, loosely coupled code. In TFD, before you start coding, you write the required tests to comply with the requirements of your software. Therefore, initially we only have the skeleton of our code and the tests. At the beginning, since the function to be tested is empty, the test fails. Then you start implementing the function. When defining the tests we provide the inputs and the right output for the function we want to test. Then, the functional code that passes the tests is implemented. In this way, developers are forced to think through the requirements and design before they start coding. The steps of TFD are shown in Figure 4.6.

TDD leads to a more modularized, flexible, and extensible code because of the testing modules built into this continuous integration can be easily modified without

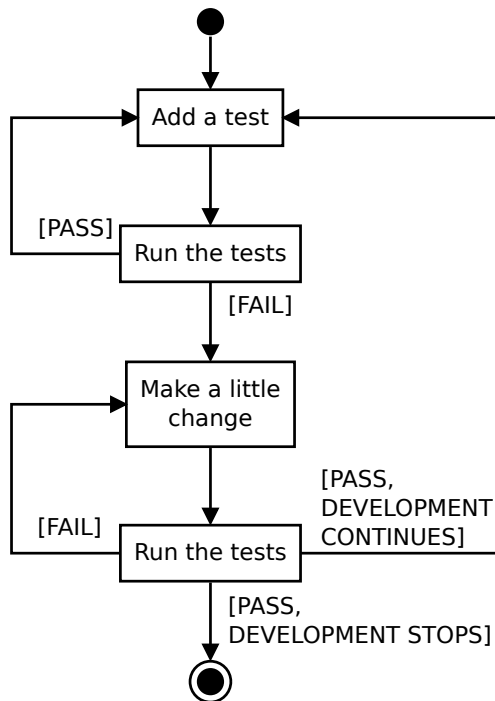


Fig. 4.6 Test-First Development flowchart.

the fear of breaking the general functionality of the code. Companies like Google or Willow Garage promotes this approach [107].

Listing 4.3 presents an example of TFD with a function that return the sum of two numbers, or an error code if the inputs are in the right format. We first create the tests for the function (Lines 6 and 7) and the skeleton of the function to be tested. In this case, the tests will fail because we just know the possible inputs and outputs, but the *sum* function is still not implemented.

```

1 function sum(a, b) {
2     return      # an empty function
3 }
4
5 # TEST
6 assert.is_equal(sum(1, 2), 3)      # fail
7 assert.is_equal(sum("1", "2"), -1) # fail

```

Listing 4.3 First step of TFD.

Now, in Listing 4.4, we have implemented the functionality of the function. Once the implementation is completed, the test should pass. For the first test (Line 10) the

result of the sum is correct, and for the second test (Line 11) the inputs are wrong formatted and the output is an error code (the value -1), so it should pass both.

```
1 function sum(a, b) {  
2     if a != int() or b != int():  
3         # error  
4         return -1  
5  
6     return a + b  
7 }  
8  
9 # TEST  
10 assert.is_equal(sum(1, 2), 3)          # pass  
11 assert.is_equal(sum("1", "2"), -1)    # pass
```

Listing 4.4 Second step of TFD.

Unit Testing

The second agile methodology we propose is *Unit Testing* (UT). A unit test is the way to check the correct working of a software module, instead of a set of functions. This is useful to be sure that every one of the modules implemented works well individually.

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct [108]. Accordingly, its main benefits are the next:

- Find problems early. Bugs are quickly found because tests only can pass or fail.
- Facilitate changes. Programmers do not fear to modify or refactoring the code because there are test to check if everything is still working as it should.
- Simplify integration. Because when integrating we are sure the code is working well.
- Improve documentation. Tests itself are also documentation for pieces of code.
- Fast design. Using tests, a test will never pass if the programmer does not implement the solution based on the design.

There are several ways to perform an unit test, but mainly depends on the programming language you use. Next subsection shows an example of how to perform a simple test.

Sometimes, doing tests is difficult because of the inter-dependencies between units. That is, in order to pass a test, a unit of code may need data from other module and

it is not present or accessible. In this situations, we propose to use **Mock** objects to simulate the behavior of modules, either hardware or software.

Mocks

Mocks are simulated objects that mimic the behavior of real objects, a hardware device or a software component, in controlled ways. Mock objects are created typically to test the behavior of a unit of code that interacts with other unattainable unit.

In robotics, this is very relevant specially when you do not have access to the robot or when developers deal with devices that are being used by others, do not work properly, or are being upgraded.

The use of a mock object also contributes to the overall modularization of the code because this allows that modules can be switched easily between the mock version for unit testing, or the real version for deployment.

As mentioned before, mock objects are very interesting to simulate the hardware level in robotics. A mock can simulate the behavior of a real device knowing the inputs and outputs for each functionality implemented.

An example of a mock object in a robot could be simulating the behavior of a touch sensor. Listing 4.5 shows the pseudocode for the *TouchSensor* class and its test using a mock. The class for the touch sensor device is implemented with a function that returns if the sensor is touched or not. When the test is executed (Line 9), the function mock function (Line 8) simulates that the sensor is touched (Line 10) and the test pass (Line 15). This test can be done without the physical device only because we are simulating its behavior.

```
1 # The class to simulate
2 class TouchSensor():
3     __is_touched = False
4
5     function is_touched():
6         return __is_touched
7
8 @patch('sensor.is_touched')
9 function test_touched(mock_is_touched):
10 mock_is_touched.return_value = True
11
12 sensor = TouchSensor()
13 status = sensor.is_touched()
14
15 assert(status == True) # pass
```

Listing 4.5 An example of a mock object.

4.3.2 Robot Operating System

Once described the software methodologies applied, we introduced here the the Robotic Operative System (ROS) framework in which we have based our implementation.

ROS is a set of libraries and tools to help software developers create robot applications [109]. Nowadays, it exists a lot of robotic projects developed with ROS. For instance, the PR2 robot by Willow Garage [110] or the NAO robot by Aldebaran Robotics [111].

ROS has several features which make itself very usable. For instance, it is *Peer-to-peer*. This consists of a number of processes in different hosts connected at runtime in a peer-to-peer topology. Also it is *Tools-based*, that is a large number of small tools which are used to build and run the various ROS components, rather than a monolithic development and runtime environment. Furthermore, it is *Multilingual*, currently supporting four very different languages: C++, Python, Octave, and LISP. Although it has its own compilation system, deep inside it works with the *CMake* system that allows for easier code extraction and reuse beyond its original intent. Finally, ROS is *Free* and *Open-Source*, this makes a large community developing at the same time and evolving robotics functionalities faster.

ROS uses code from numerous other open-source projects, such as the drivers, navigation system, and simulators from the *Player* project, vision algorithms from *OpenCV*, and planning algorithms from *OpenRAVE*, among many others.

The main ROS concepts that we use in our system are metapackages and packages, nodes, topics, messages (publisher/subscriber), and services (server/client). The general scheme of a ROS based system could be found in Figure 4.7.

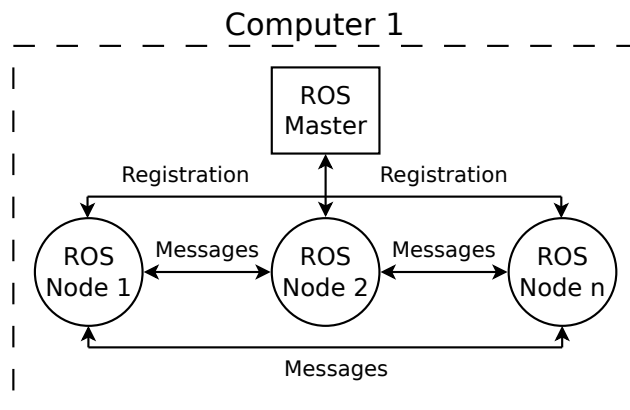


Fig. 4.7 ROS general scheme.

Firstly, a **package** is a software project where functionalities are implemented in a node. Furthermore, a *metapackage* does not have any source code because it only has dependencies to other packages, that is, it is like a container which references to packages it includes. Another important concept is **nodes**, which is an executable that uses ROS to communicate with other nodes. Nodes can *publish* messages to a topic as well as *subscribe* to a topic to receive **messages**. Nodes can also provide or use **services**.

Messages are ROS data type used when subscribing or publishing to a *topic*. A publisher send a message when data has to be transferred to another node. Meanwhile, other node, that is subscribed to the topic, will receive the message.

Services are another way that nodes can communicate with each other following a server-client paradigm. Services allow nodes to send a request and receive a response synchronously. The server node provides a service that will receive the client call, realize the operations, and return the result. A client will do a petition on demand.

Both last concepts could be seen graphically in Figure 4.8.

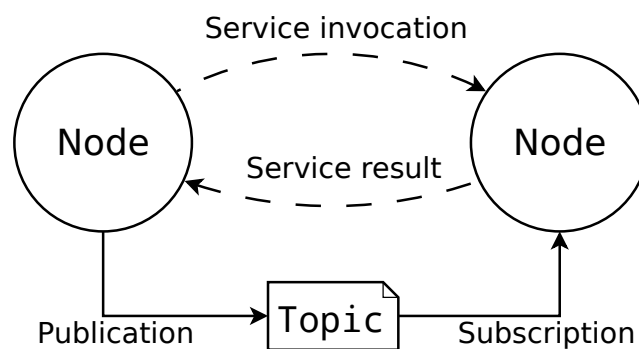


Fig. 4.8 ROS communication between nodes.

Another important tool ROS offers us is the **actionlib** stack³. The actionlib stack provides a standardized interface for interfacing with preemptable tasks. Examples of this include moving the base to a target location, performing a laser scan and returning the resulting point cloud, detecting the handle of a door, etc.

In any large ROS based system, there are cases when someone would like to send a request to a node to perform some task, and also receive a reply to the request. This can currently be achieved via ROS services.

In some cases, however, if the service takes a long time to execute, the user might want the ability to cancel the request during execution or get periodic feedback about

³Actionlib. ROS wiki. Link: <http://wiki.ros.org/actionlib>

how the request is progressing. The `actionlib` package provides tools to create servers that execute long-running goals that can be preempted. It also provides a client interface in order to send requests to the server (Figure 4.9).

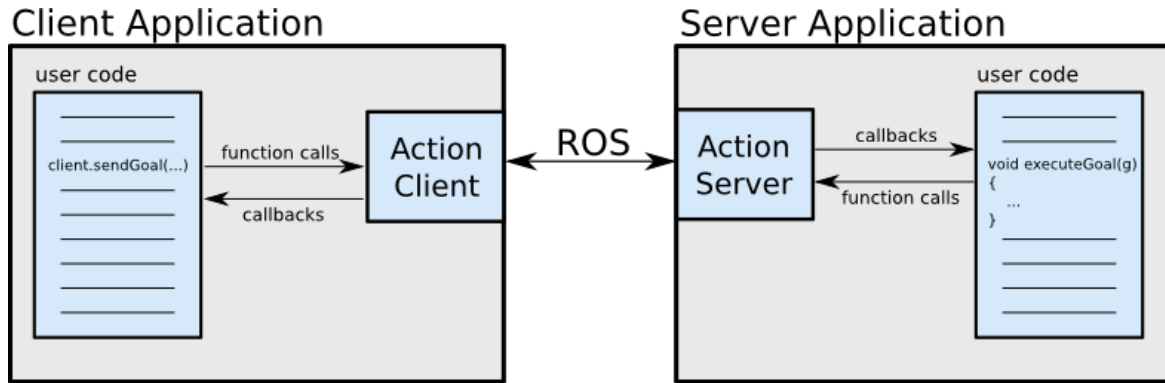


Fig. 4.9 The ActionClient and ActionServer communication.

In order for the client and server to communicate, we need to define a few messages on which they communicate. This is with an action specification. This defines the *Goal*, *Feedback*, and *Result* messages with which clients and servers communicate:

Goal To accomplish tasks using actions, we introduce the notion of a goal that can be sent to an *ActionServer* by an *ActionClient*. An example of goal would be for controlling the tilting laser scanner, the goal would contain the scan parameters (min angle, max angle, speed, etc).

Feedback Feedback provides server implementers a way to tell an *ActionClient* about the incremental progress of a goal. As example for controlling the tilting laser scanner, this might be for instance the time left until the scan completes.

Result A result is sent from the *ActionServer* to the *ActionClient* upon completion of the goal. This is different than feedback, since it is sent exactly once. This is extremely useful when the purpose of the action is to provide some sort of information. For the tilting laser scanner controlling example, the result might contain a point cloud generated from the requested scan.

We have defined a common result definition where the consequence of an action may only be **SUCCESS**, **FAIL**, or **ERROR**.

4.3.3 Implementing the low-level components

In this layer the main goal is to achieve the abstraction of the hardware and simplify the information for the high-level. At this level, the software modules dealing with the hardware are located. For each device existing in a robot, we propose the different modules presented in the UML class diagram shown in Figure 4.10. In this diagram we show three main parts. The first is the external code the manufacturer gives, usually called third party code. This library could be used by one or more wrappers. In the second part, an inheritance between the interface class and the drivers, wrappers, and mocks. Finally, the node to control the device based on the interface functionality is shown. This node could use one or more interface objects depending on its necessities. Besides, it will content the services to offer to the high-level layers.

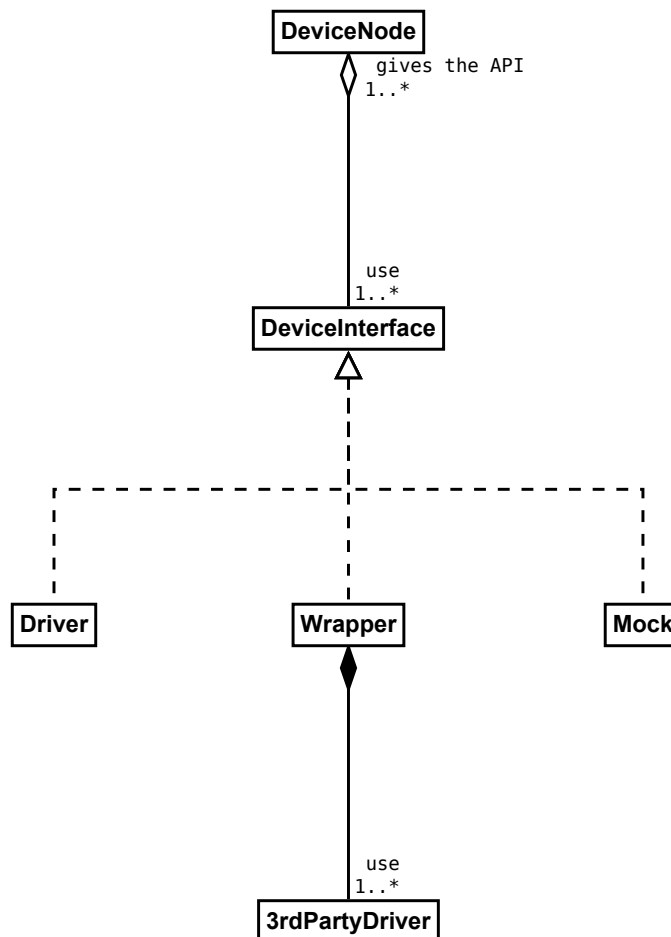


Fig. 4.10 UML class diagram for low-level components.

The Robotic Architecture

This design is made based on the **Interface Pattern** design. As we commented in Section 4.3.1, IP offers to this design the advantage to be easily developed and quickly tested. In addition, the main advantage we have is the possibility to add, modify, or remove the low level software for a device in an easy and fast way. This kind of class interface design allows to simplify the complexity in the low-level software for the hardware of a robot.

A common practice in robotics is the integration of multiple devices from different vendors. Some of these devices come with its own software (a library) to control them and developers have to integrate it into their systems. Others provide the protocol to communicate with the device and the driver has to be developed from scratch. In the first case, we suggest to create a *wrapper* that uses the third-party library and implements the interface. In the second case, the *driver* has to implement the functionalities described in the *device interface*.

Once the driver or wrapper is implemented, we focus on the devices. Each device in the robotic system provides its capacities following the client-server paradigm. One device is accessible through a *node* (i.e. a software module) that implements a server that can be used by other nodes in the higher level. These devices can be sensors or actuators of a robot, for instance the touch sensors in the body, the lasers for navigation, a robot's eyes, or the motors for the wheels. We think devices should only offer services to the upper levels. This is because devices could overload the system if the status of an actuator or every data read by a sensor is published continuously. Offering services, the higher levels could control the rate for reading or writing data and also could detect status changes publishing only the data desired. Furthermore, through service communication high layers can get the information on-demand.

We apply the **Dependency Injection** pattern in the device nodes. When creating one of these nodes, developers pass the driver dependency through constructor argument. This offers the possibility of having several driver implementations for the same kind of device in different robots, and it is transparent for the rest of modules in the software architecture. For instance, if we have two robots with different RFID readers, we can reuse the *RFID Node* and the *RFID Interface* because they will provide the same services in both robots (e.g. read a rfid tag), but there will be two drivers or wrappers, *RFID A* and *RFID B*, with its particular implementation. When the *RFID Node* is created, developers pass the dependency to *RFID A* or *RFID B*, depending of the robot.

Moreover, following the **Unit Testing** approach, each device will have a *mock* that emulates the behavior of the device. The *mock* has to implement the functionalities

defined in the *Interface* too. Consequently, using *mocks*, testing can be done without the hardware and developers can check the software without the physical device connected.

In this way, we have a decoupled implementation of devices where the functionality is still being the same even replacing the physical device by another. Hardware designers could add, update or remove devices for a robot as many times as they need without affecting the high-level functionalities or communications.

All these features provide the advantages of making the low-level software for a robot modular and easily adaptable in the usual case of devices substitution, or when multiple people are developing at the same time for one robot.

Before focusing in the detailed explanation for a concrete device, we present what packages are in the robots, Maggie and Mini. As we commented in Section 4.3.2, a metapackage named as `<robot_name>_devices` contains the references to the packages related. For the robots, Figure 4.11 shows in a UML package diagram the structure followed. There is also a *common_devices* package that shares the common devices for both robots.

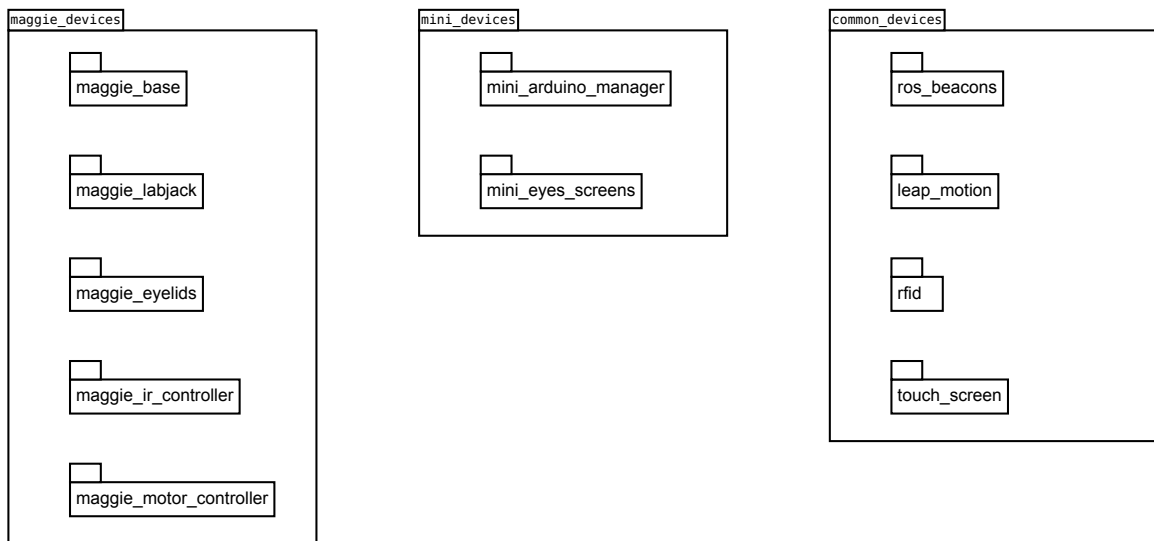


Fig. 4.11 UML package diagram for Maggie and Mini devices.

All devices follow the methodology presented in Section 4.3.1. To illustrate, we show the case of the Labjack device (Figure 4.12), an input/output DAQ used to control elements such as touch sensors, batteries voltage, and other digital and analog signals. For this case, the manufacturer provides its own driver library (the *ljacklm* driver) but it has to be integrated in the system. Following our proposal, a wrapper

The Robotic Architecture

class (*ljacklm_wrapper*) is developed in order to integrate such functionalities provided by the manufacturer in a common interface.

Additionally, we implement a mock (*mock_ljacklm*) to simulate the Labjack functionalities. This mock offers the possibility to emulate the Labjack device without having it physically in order to test the device at any time. In addition, every class generated has their unit tests included, making easier to generate new pieces of code without losing the main functionality.

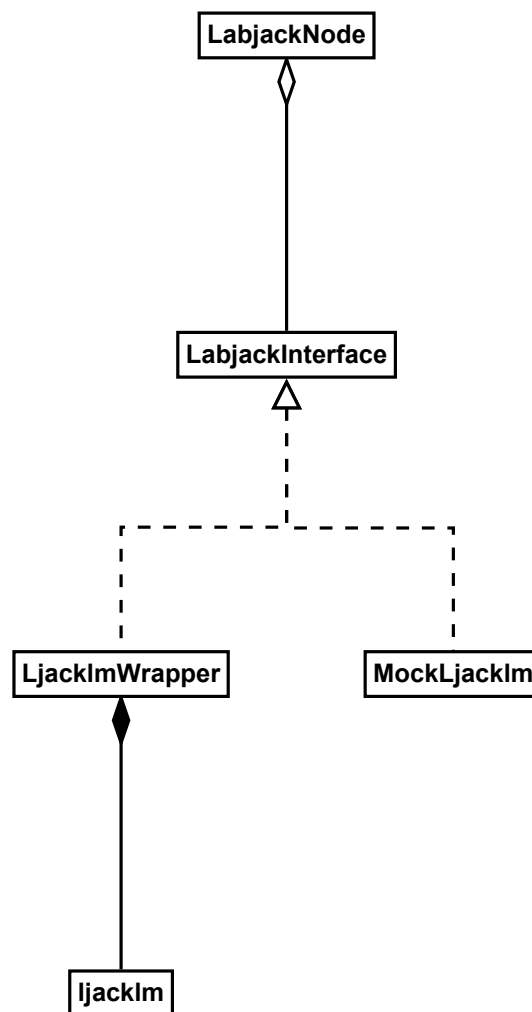


Fig. 4.12 UML class diagram for the labjack device.

We have used the same approach for the device nodes for communicating with the high-level. That is, ROS services (servers/clients) that are always available waiting for a specific petition. The most of these nodes have been implemented in the C++

programming language considering that the vast majority of third party hardware drivers are implemented in this language - commonly in C or C++.

Therefore, using the TFD methodology explained in Section 4.3.1.2, the best practice is to implement the unit tests and mocks before programing the functionalities of the nodes. In our particular case, at this level we have used the Google Test [107] and Google Mocks [112] libraries for implementing unit tests in C++.

In order to a better understanding about unit testing and how to apply it, we provide an example for a specific functionality of the labjack device. Listing 4.6 shows the functionality developed. This function implements a read of an input in the labjack device which get information about if the robot is in the emergency state. This function will return a zero value if everything is correct, or a minus one value if the driver returns an error.

```
1 int LabjackNode::isEmergency()
2 {
3     int result = 0;
4     long emergency_line = IO_EMERGENCY;
5
6     result = getStateIO(emergency_line);
7
8     return result;
9 }
10
```

Listing 4.6 Function of the labjack device.

The unit tests implemented are shown in Listing 4.7. These tests show first the configuration of the mock driver to return what we expect. And later, a labjack node with the driver passed as *dependency injection* in the constructor called. Finally, we test the implemented function in base of its outputs. The test will pass or fail depending on the output, simulating here the answer of the driver.

```
1 // Test isEmergency()
2 TEST(LabjackNode, IsEmergency)
3 {
4     MockLjacklmWrapper mock;
5
6     EXPECT_CALL(mock, config())
7         .Times(1)
8         .WillOnce(Return(0));
9     EXPECT_CALL(mock, readDIOs(_, _))
10        .Times(1)
11        .WillOnce(Return(0));
```

```
12 EXPECT_CALL(mock, readAIs(__, __, __))
13   .Times(2)
14   .WillRepeatedly(Return(0));
15 EXPECT_CALL(mock, readDirectionDs(__))
16   .Times(1)
17   .WillOnce(Return(0));
18
19 LabjackNode labjack_node(&mock);
20
21 // success
22 EXPECT_CALL(mock, readDIOs(__, __))
23   .Times(1)
24   .WillOnce(Return(0));
25
26 ASSERT_EQ(0, labjack_node.isEmergency());
27
28 // fail
29 EXPECT_CALL(mock, readDIOs(__, __))
30   .Times(1)
31   .WillOnce(Return(-1));
32
33 ASSERT_EQ(1, labjack_node.isEmergency());
34 }
35
```

Listing 4.7 Unit test for a function of the labjack device.

4.3.4 Implementing the high-level components

Once the low-level layer of our architecture is developed, the skills are able to communicate with the devices implementing more complex behaviors. For instance, an example of skill could be one assigned to control the changes in the level of battery more than the voltage. That is, depending if the batteries are critical or full, through calling the services of the corresponding device node, this node will be able to publish that state changes in order to make a decision or another.

Figure 4.13 shows the main structure of how skills and devices are related and how they communicate between both. Using the common interface classes, skills are able to communicate with devices through their services. Moreover, skills provide the specific messages publishing the information about relevant changes (i.e. a person is detected, the robot is running out of battery, or the robot is lost).

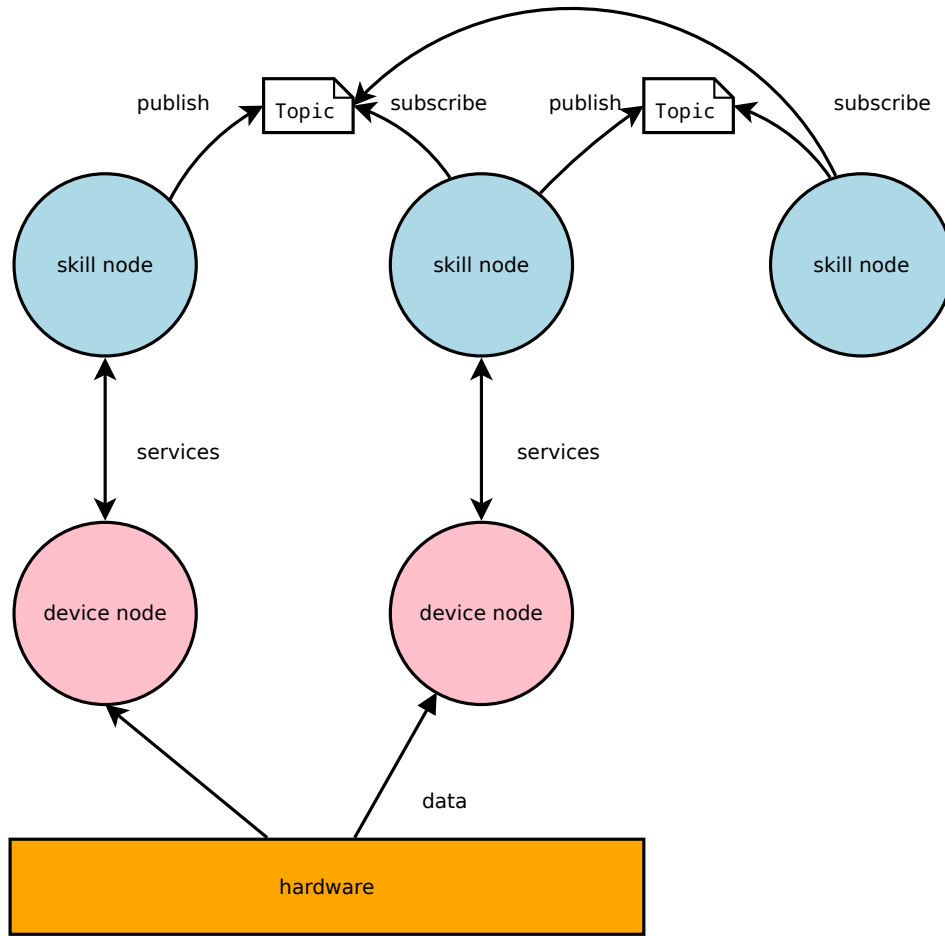


Fig. 4.13 General scheme for some skill nodes depending on devices.

In this way, the implementation for the skills of the robots are quite different with respect to the devices. Trying to provide modularity and portability for the any robot, some of the skills we have developed perfectly works in both robots. Figure 4.14 shows the UML package diagram for the skills implemented. We have some specific skills that are continuously running a control loop such as the basic states, the batteries, the touch, and the blink skill packages - *continuous skills*. On the other hand, we have other skills that are executed by the decision-making system on-demand such as the included in the *common_skills* package - *conditional skills*. It has to be pointed out that there are some robot specific skills such as related to navigation that do not fit both robots due to the limitation of capabilities.

At this point, we can present a real example for a skill implemented in order to show the structure followed. Analogously to Figure 4.13, the Figure 4.15 shows a

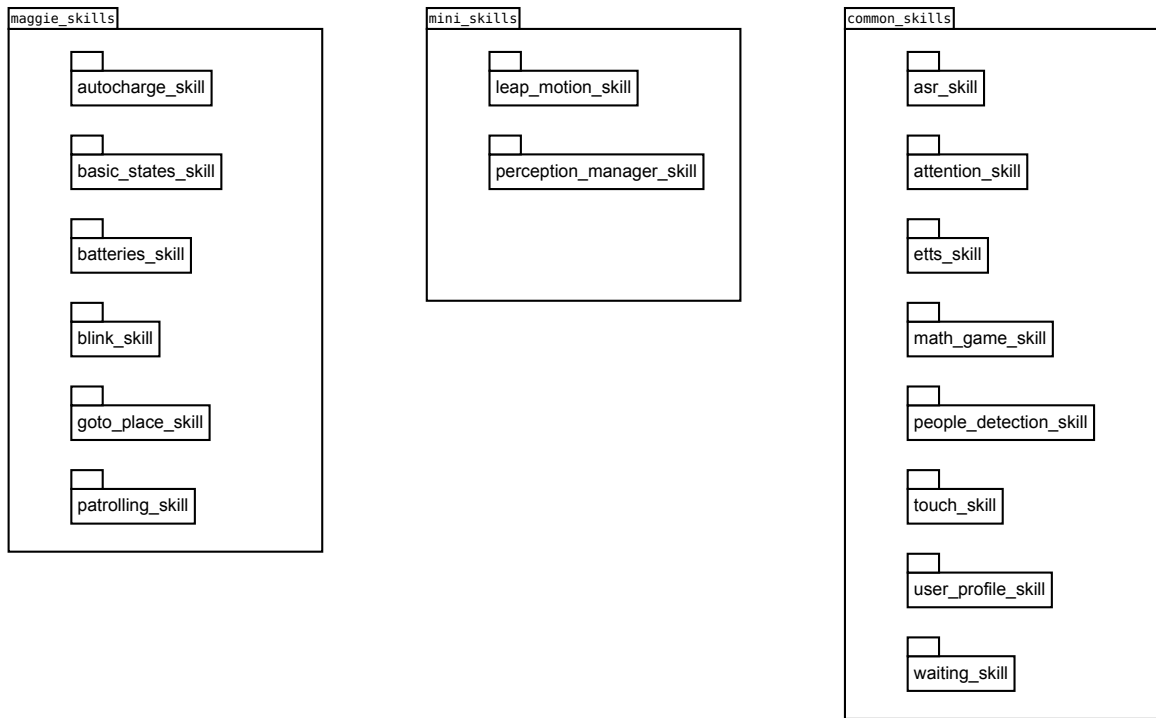


Fig. 4.14 UML package diagram for common skills.

complete diagram for the touch skill and its dependence on the labjack device. For communicating, the labjack node device has a server in the low-level always waiting for a petition of any skill. The server is available with the topic called *labjack/is_touched* providing information about all sensors in the body of the robot with a structure of array in which the content is zero for sensor not touched and one for touched. In the high-level, the touch skill node has a client connected to the same topic that is launched every time there is a petition. In the main loop, the skill is asking periodically for the sensors touched. Once the array is received, it is converted in a more human readable format publishing only the information about the changes. The published information is done in a topic called *touch_skill/is_touched*. Thus, the decision-making process could have a drive for instance that could evolve depending on the state of the touched sensors. Section 5 will explain in detail the skills used in the experiments.

To test every device and skill is working properly, we have developed a teleoperation GUI following the MVC paradigm explained in Section 4.3.1. Through this graphical interface a controller is able to see the status of the robot, send commands for motor teleoperation, navigate to specific points in a map, among others. Although autonomy

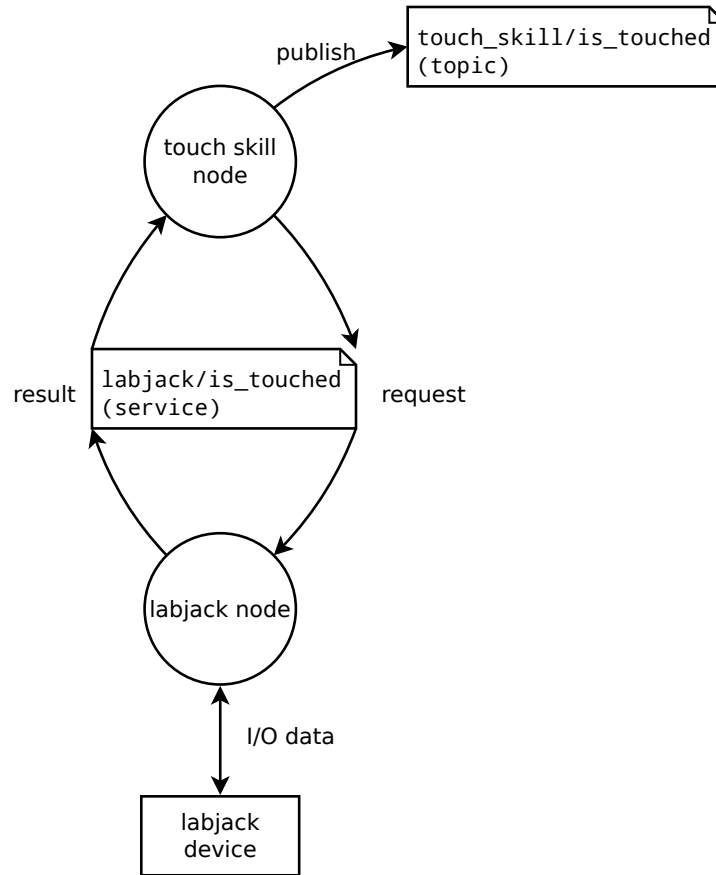


Fig. 4.15 Diagram for the touch sensor skill.

is the final goal of this robot, this kind of GUIs are very useful to test and debug at any moment of the life of the robot.

For this level layer, the predominant programming language we have used is Python instead of C++. This is due to Python language usually offers better high-level capabilities to facilitate and make powerful skill implementations. Because ROS and the *actionlib* library are compatible with several programming languages, there exist the possibility of programming other skills in the language preferred by the developer. Additionally, we have used the *unittest* [113] and *rostopic* [114] frameworks to implement the unitary and integration tests on this layer. These frameworks work similarly to the Google libraries but integrated with Python.

4.4 The Decision-Making System

This section is one of the most important to the work developed for this thesis. Here, the implementation about the decision-making process is fully described. For every step of the process, we present how it is developed and the technologies applied.

Before presenting every component by separately, we provide a complete description about the system developed showing their main modules or nodes (Figure 4.16). Our system is based in four main components, each one related to an important part of the decision-making process.

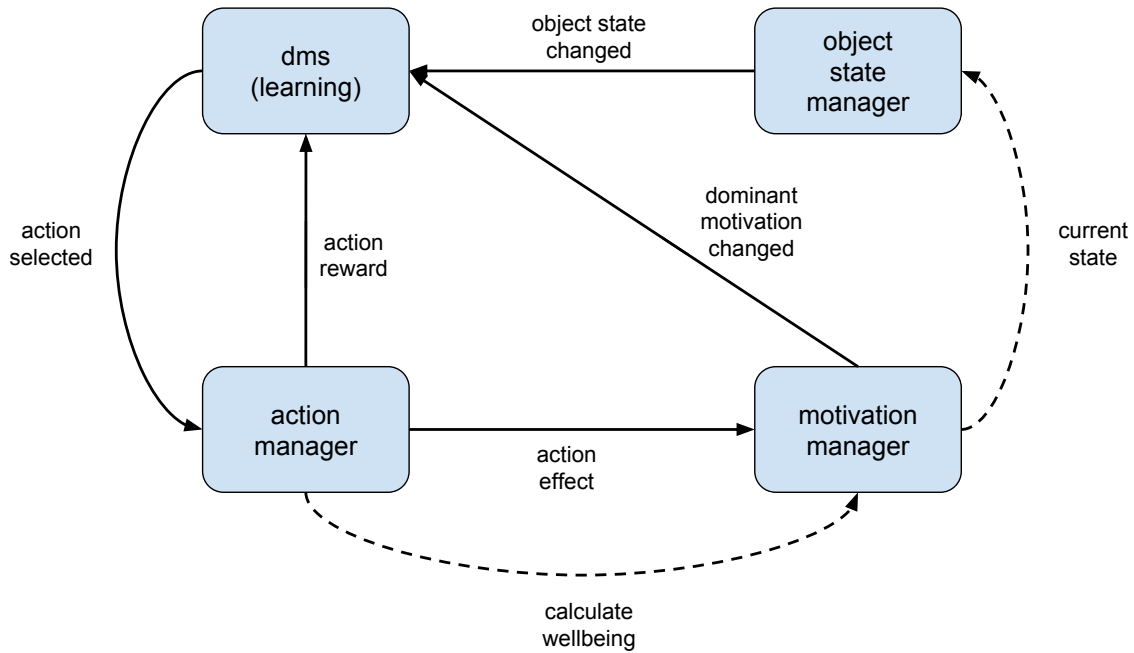


Fig. 4.16 DMS components flowchart.

Additionally, for a better understanding of every component in the DMS, we show to the reader relevant information about how data is stored. Almost every parameter the DMS uses is stored in a database that models the information for setting up these components. The database will also store information about communications in ROS, for instance the topic related for a user or the topic and type of data an action received as a goal. All these features are completely described next when presenting the module responsible of the related data. Thus, Figure 4.17 shows the database diagram made by the *MySQL* service in which is represented the entity-relationship scheme.

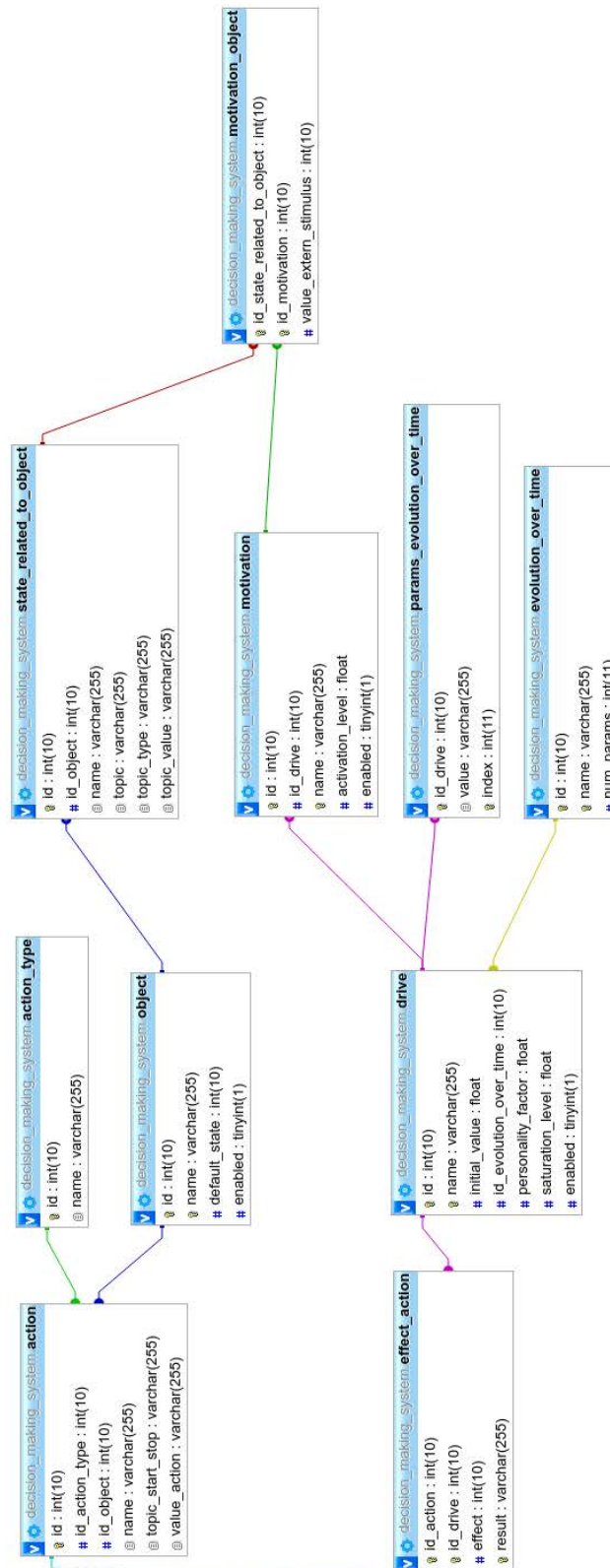


Fig. 4.17 Database diagram for components.

4.4.1 The States Managers

Firstly, we focus again on the state of the robot. We have developed a real-time states monitoring, where internal and external states are communicated to the DMS every time a change is detected. We have implemented those modules following the class structure defined in Figure 4.18. For every object (or user) there is a related state that can act as an external stimulus for a motivation. Besides, there are actions that are associated to a specific object. Those actions always have an effect affecting a drive that is evolving over time.

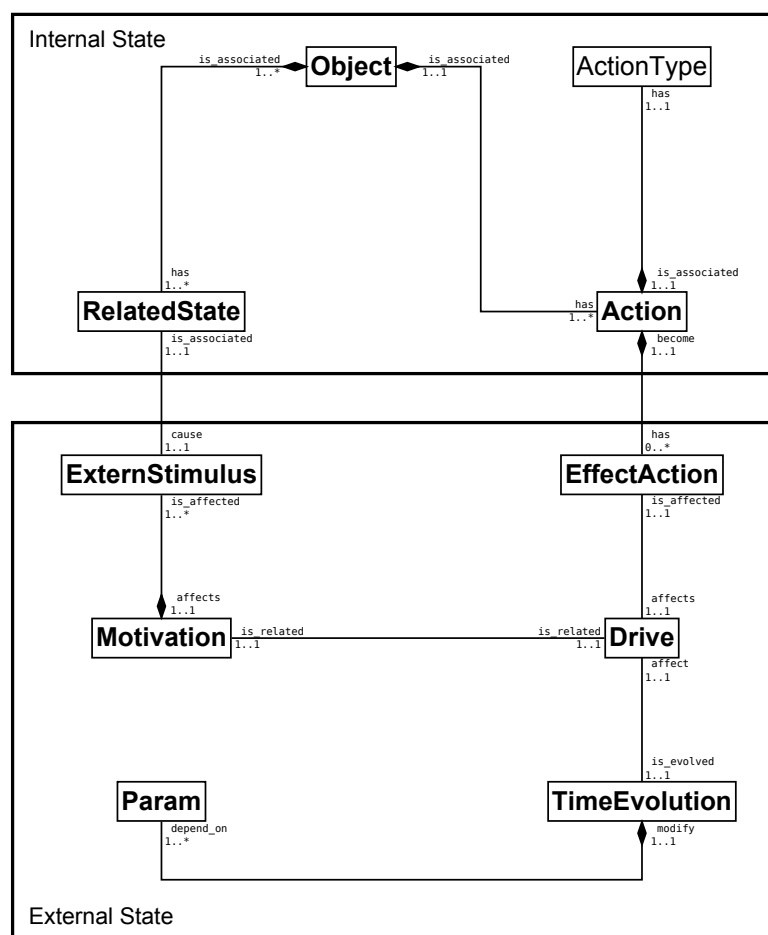


Fig. 4.18 UML class diagram modeling the internal and external states.

The external state of the robot is modeled from the states of the people who is going to interact with it. The process of monitoring people has been developed in two phases depending on the place of the user in the environment. Figure 4.19 shows a perception module that monitors if a user is *present* or *absent*. In addition to that, we have a face detector perception module to recognize where a user is in front of the robot ready for *interacting*. Nevertheless, these two modules are based on perception. Perception usually has some limitations. For that reason, these modules are evolving continuously in order to improve the accuracy of detectors. In this way, we have modularized the people monitor node so it can subscribe from any detector to improve user's states detection.

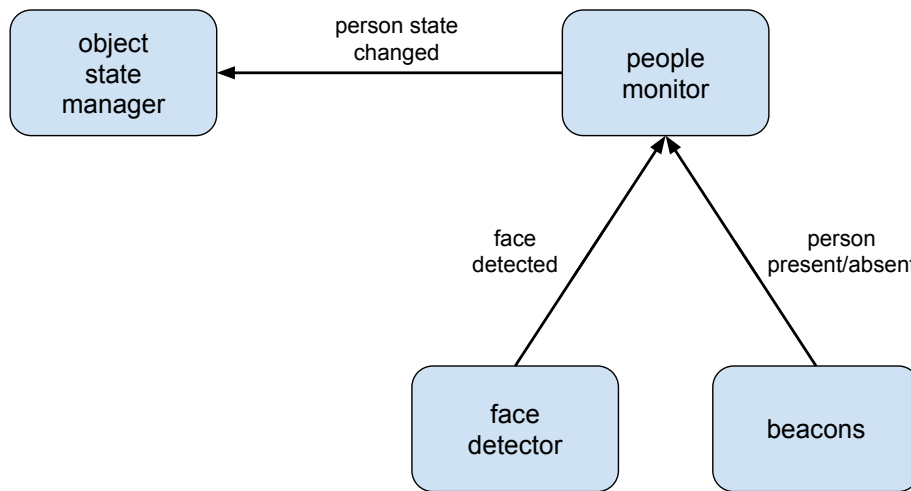


Fig. 4.19 People state monitoring.

The **object-state manager** is a ROS node that has a publisher. This publisher sends the information related to a change in the state of a user. We have included here a multi-subscriber for every object, an innovative structure implemented in Python and ROS that generalizes the communications simplifying the process for future detectors. That means that every time a user changes its state, the related subscriber will be listening for updating the information about the external world. In addition to that, a service server has been included for requests about the information about the state of a specific user.

As we commented before, we only consider people as the objects in our environment. Thus, the **people monitor** node has been implemented for monitoring the current state of those active users in the environment of the robot. This node also includes

a publisher for the active user that is in the state *interacting*. This publisher will facilitate information about the user that is going to play the game.

Detecting if a user is present or not in a room is anyway a difficult perception task. For this thesis, we have chosen a novelty technology based on *beacons* - a short range Bluetooth emitter. With this kind of device, the robot is able to know if a user is present or absent depending on the distance to itself. We have considered that a distance of 10 meters is enough to set the user's state to present. Thus, if the beacon sensor is near enough, the user's state will change to the *appearing* state. However, if the sensor is in a distance considered as far, the transition will be to the *disappearing* state.

However, interactions with the robot are made in a short distance. So, we needed a more robust procedure to detect if a person was in front of the robot or simply near. For that purpose, a face recognition is included only for people detection. In this case, an RGBD camera (*Microsoft Kinect*) is used to do the face analysis. Once a face is recognized for a user in the *near* state, that user will change the state to the *approaching* state. When a user is in the interacting state, if the face is lost by the detector the state will change again but now to the *leaving* state.

The states for a user has been modeled in the database as shown in the example in Table 4.1. The table saves the name of the topics related to each known user. Moreover, it is stored the type of message that can be sent, e.g. *String* or *Bool*. Lastly, the value of the state sent is defined as the same as the name for the state.

State Name	Topic	Topic Type	Topic Value
absent	people_monitor_skill/Raul	String	absent

Table 4.1 States stored in the database.

To implement the **motivation manager**, we have developed a ROS node that is controlling in real-time which is the *dominant motivation* and publishing to the DMS module if there is a change. This node is also responsible for launching every *drive* defined in a parallel thread that is evolving continuously. In order to update the drives when an action has been executed, a subscriber is included checking the action effect. Moreover, this node acts as a server service for the wellbeing of the robot, being possible to return the value for the current wellbeing whenever it is asked.

Once the robot has defined its state, it is the moment of deciding what action to launch. Figure 4.20 shows the package diagram with the classes included for

every component in the database and the learning components. This architecture benefits the structure and organization for all components developed. Through this software methodology, as the same of the devices and skills, we anticipate the inclusion of future implementations adding tests for each module developed. Besides, the modular structure provides an easy way to add other learning methodologies for future improvements.

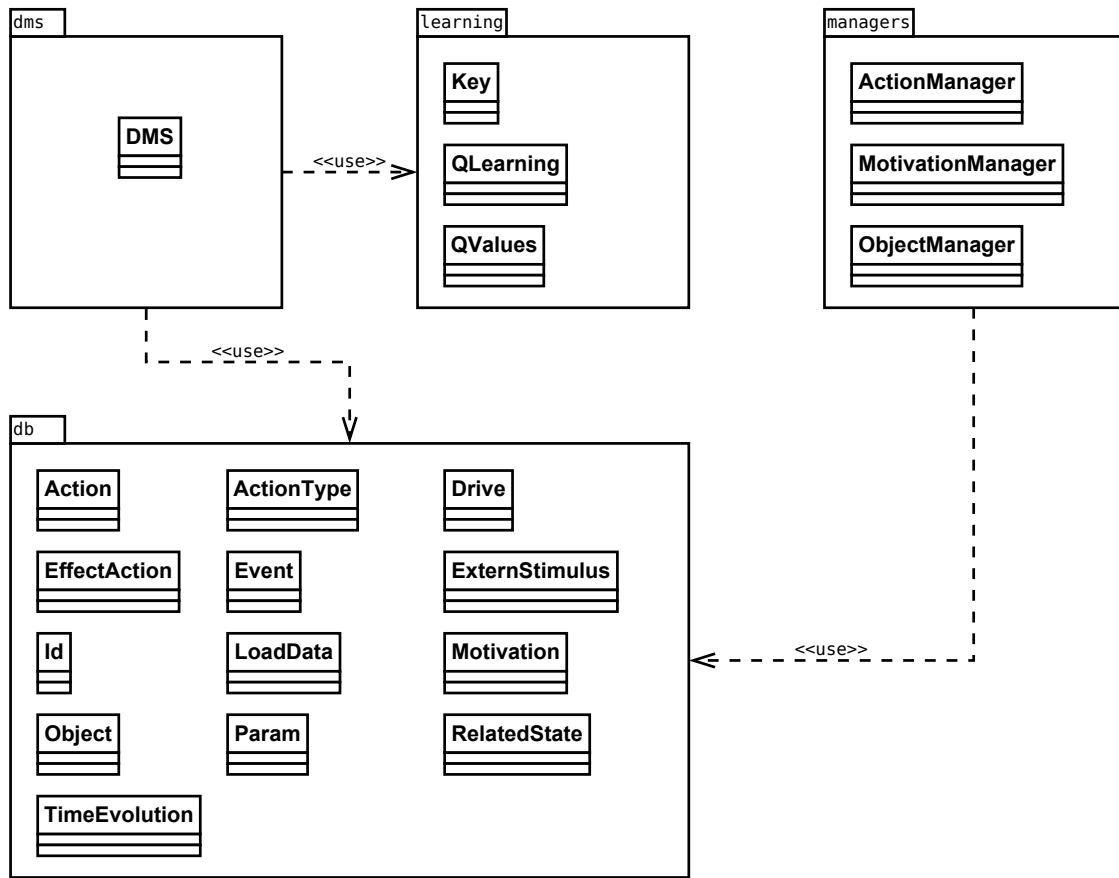


Fig. 4.20 UML package diagram for the DMS.

4.4.2 The DMS

The most important node implemented is the DMS. The DMS is the responsible of deciding what action to execute at any moment. Furthermore, it implements the learning methodology included in this thesis that update the Q-Values for any *action-state* pair.

The DMS node contains two publishers and three subscribers. One publisher is for the action selected. As we explained in Section 3.5, the action is selected following a probabilistic technique that includes the knowledge learned in previous executions. When a change in the internal or external state (one subscriber per manager) is received from the managers, the evaluator is called to reevaluate what is the best action to do with those conditions. In the case that there was a previous action executing, the evaluator sends an interruption signal, the second publisher included, to finish that action as soon as possible and to execute the new one. Thus, the evaluator does not send the new action selected until the last action has finished. Once the action has finished, either naturally or by an interruption, a subscriber is listening for the reward of that action.

It is in this moment when the learning process starts. The Q-Learning algorithm is executed with the current information the robot has. This information is: the dominant motivation when the action started, the dominant motivation when the action finished, the user and the state before and after the action execution, and the reward of the action. Finally, the evaluator is called again to select a new action although there is no change in the state.

4.4.3 The Action Manager

Before describing the action execution process, Table 4.2 shows an example about how actions have been modeled in the database. It can be found here the topic included to start and stop the skill associated to the action. Every action has also to have an associated input value that modify the behavior of the skill such as the level of difficulty for a game or the type of communication in a conversation. This parameter could be also empty.

Action Type	User	Action Name	Topic start/stop	Value
endogenous	Raul	nothing	waiting_skill	300
exogenous	Raul	attract_attention	attention_skill	greet

Table 4.2 Actions stored in the database.

Once the action is selected, the **action manager** node will execute the action until it finishes or is interrupted. The *action manager* node has two publishers, one for

the action effect and the other for the action reward. Besides, it has two subscribers receiving information about the action selected by the DMS and a signal if the current action has to be canceled.

The *action manager* follows common steps for every action selected by the DMS. Get the initial wellbeing. Publish to the skill selected the start signal. The client handler send the goal to the skill. This waits until the server is ready, then send the goal that is the value obtained from the database. While the skill is executing, the *action manager* waits for the result. Once the result is received, the skill is stopped. After that, the *action manager* applies the action effect publishing the information for the DMS. In this moment, the final wellbeing is obtained from the *motivation manager*, and the reward is calculated and published.

If no cancel signal is received, the skill will execute normally doing the action or actions related. Otherwise, if a cancel signal is received and an action was executing in that moment, the action client handler will transmit the cancel signal to the skill doing this what it should.

Therefore, we have also defined common steps for creating a new skill. These are:

1. *Set the type of skill.* Depending if there is a continuous or conditional one.
2. *Create the heritage from skill.* Include the name of the new skill and its type.
3. *Define communications.* Depending on our package dependencies you will need some publishers, subscribers, servers or clients. Declare them to the “create_msg_srv” function. If you are defining a conditional skill, create the actionlib server setting up the “execute_cb” and the actionlib message. Do not forget to close communications in the “shutdown_msg_srv” function.
4. *Implement the main procedure.* This will be a different process depending on the type of skill.

If conditional:

- (a) *Define interruptions.* Before implementing the execution callback in conditional skills, define clearly the point break interruptions for the procedure. This step is not needed for continuous skills.
- (b) *Return feedback.* Return some feedback about the execution, this is optional.
- (c) *Set SUCCESS, FAIL, or ERROR.* Depending on the action execution, the result may be a success or a fail. If the skill receives a goal while not active (RUNNING), the result must be an error.

- (d) *Return result.* Return the result for the action completion.

When the action finishes, the effects associated to an action are modeled following the *Actionlib* configuration where the result of an action can be ERROR, FAIL, or SUCCESS. Table 4.3 shows an example for an action that have one effect on a drive depending on its result.

Action	Drive	Effect	Result
show_question_easy	user_satisfaction	-10	SUCCESS
show_question_easy	user_satisfaction	0	FAIL
show_question_easy	user_satisfaction	10	ERROR

Table 4.3 Actions effects stored in the database.

In order to keep all skills communicated with the devices, the *working memory* of this architecture is developed as a ROS functionality based on a blackboard pattern system called the *parameter server*. This kind of memory has some differences compared with the theoretical description about the key properties commented in Section 3.6.2. That is, in this implementation there is no limitation about the number of chunks the memory can store. However, information is always shared for all processes being accessible at any moment. Besides, the content stored can be updated and manipulated by any component, being eliminated whenever the system needs it.

Finally, for the skills related to HRI we have developed an *episodic memory* modeled as a database where the skill can manipulate, update and remove the information about past interactions with the users. Figure 4.21 shows the database entity-relation diagram.

The information that is managed by this memory refers to the known users that are able to interact with the system. For a specific user, some personal information is stored - the name and the age. It is also stored the number of interactions and the date and time for the last interaction. Additionally, a list of all available games is stored in order to save information about if the user has played such game and the last time played.

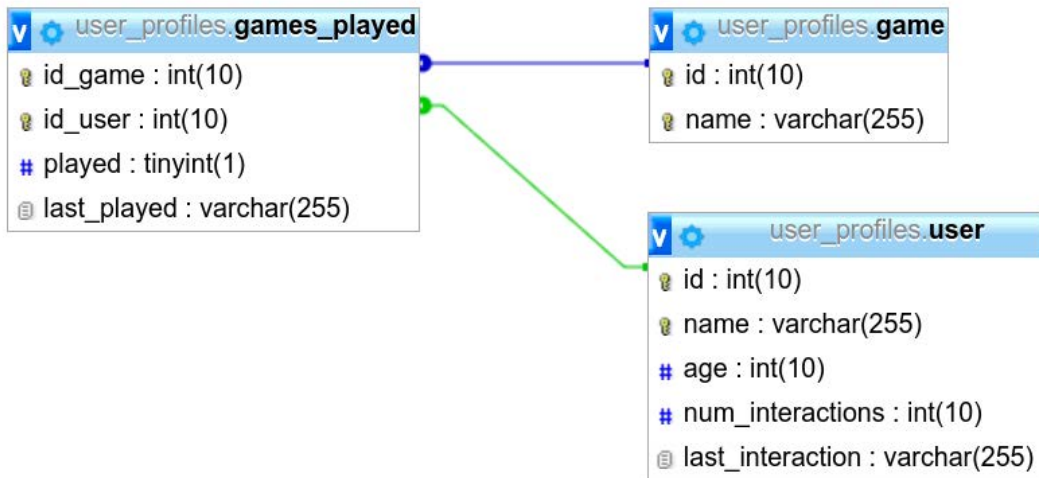


Fig. 4.21 Database diagram for users (episodic memory).

4.5 Conclusion

This chapter has presented the previous works related to the implementation developed. Including the software design and the implementation of the robotic architecture together with the proposed decision-making system for Socially Interactive Robots. The aim of the chapter has been to establish the technologies applied during the development of the system. The DMS has been included as one more module in the architecture working perfectly in ROS and applying well-known software methodologies. These methodologies has provided us with real benefits, such as modularity and portability, when deploying the architecture in different robots with the objective of interacting with people.

The complete system presented in this chapter is evaluated following in a series of experiments to demonstrate the system is working properly. The SIR are able in this moment to make autonomous decisions depending on its internal and external states, while a continuous learning is involved in every interaction with the users.

Chapter 5

Experimental Results

No amount of experimentation can ever prove me right; a single experiment can prove me wrong.

Albert Einstein

5.1 Introduction

After designing and implementing the robotic architecture and the DMS, the experimental setup is described in order to evaluate the performance of the system.

These experiments are focused on evaluating the performance of the DMS taking into account the learned behaviors. These behaviors will depend on the conditions where the robot has been learning. Since this thesis is directed to HRI, our DMS will handle the actions of the robot while interacting and depending on the different user profiles. In this case, the environment is specifically defined for HRI. Users are the only object the robot is able to interact and to monitor their states.

The learning process is one of the important concepts that we try to evaluate in this thesis. For that reason, we have carefully designed a experiment where the main objective is testing if the robot is able to achieve a long-term interaction.

5.2 Our Socially Interactive Robots

In this section we present the robots that have been used during the development of this thesis and later for the experiments. For each robot, the more relevant hardware components are listed to understand the actions the robot will be able to do.

5.2.1 Maggie

Maggie [115] (Fig. 5.1) is a mobile social robot created to general purpose services, such as interact with the environment or people around it.

Maggie is a multifunctional robot that has developed several applications throughout his life. More concretely, this robot can be used like: entertainer, singer, dancer, chatter, or health assistant among others [84][116].



Fig. 5.1 Social Robot Maggie.

In relation to its hardware components, Maggie has suffered a lot of modifications over years. The robot started implementing basic functionalities as a mobile robot. But little by little, this robot has been upgraded with new sensors and actuators to improve its action repertoire.

Currently (Fig. 5.2), Maggie is a computer-controlled system built on a wheel base which allows the robot move through the environment. From top to bottom, Maggie has in its head a motorized eyelids that can be moved in a human-like manner and a touch sensor in the upper part. In the nose, an RFID reader is placed for identifying “objects” such as identification cards or medicines. In its mouth, an illuminated vumeter shows when it is talking in addition to a webcam used to people detection and

5.2 Our Socially Interactive Robots

identification. All these parts of the robot in conjunction can show some expressiveness when interacting with people.

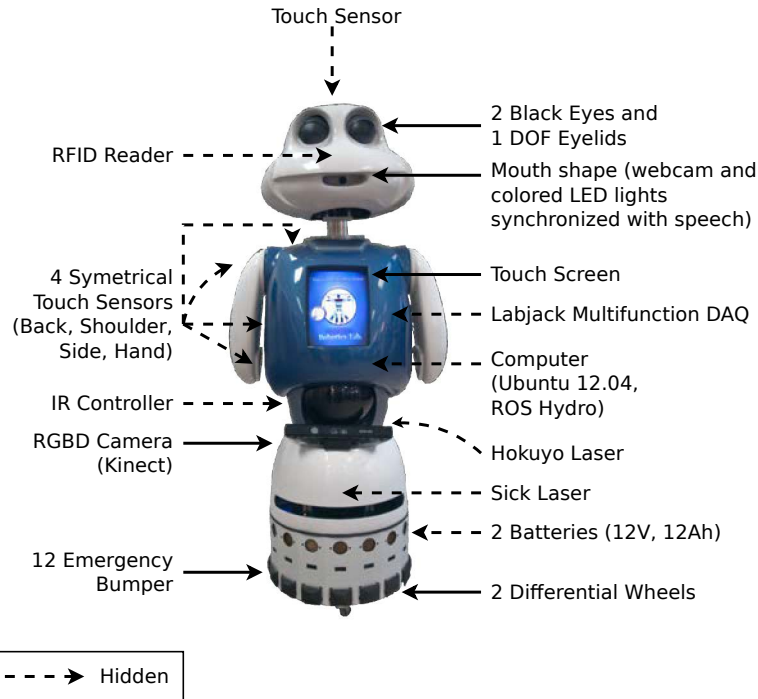


Fig. 5.2 Hardware architecture for the social robot Maggie.

Next part we can find is the neck followed by the torso. The neck is a motorized articulation with two degrees of freedom (pan and tilt). And the torso is a fixed part of the body where we can find externally two motorized arms and a touch screen in its chest. The touch screen can be used to show videos, images, or even menus for interacting in games or exercises. Besides, in the inner we can find the main computer, an acquisition card for sensors, several touch sensors distributed in the arms and the rest of the torso, and a IR controller to control a TV.

In the zone between the torso and the base, we can find in the front part an RGBD camera (kinect) used for navigation and people detection. In the back part, a hokuyo laser is placed for navigation applications such as a wall finder to detect the dock station.

Finally, a mobile base (similar to a pioneer 2 robot) with two differential wheels is placed in the bottom of the robot. In this part, we can also find a sick laser for navigation purposes. The robot has twelve emergency bumpers around it in order to preserve the security of the robot if there is a collision.

5.2.2 Mini

The second robotic platform used is Mini [117] (Figure 5.3). This robot is a desktop social robot created to interact mainly with elderly people or kids. Although Mini is mainly planned to help elderly people with some kind of disability such as dementia or Alzheimer, there are many applications this robot is able to do. Between them we can find: dancing, have conversations, play educational or entertaining games, play cognitive exercises, and even do limited physical rehab.



Fig. 5.3 Social Robot Mini.

In this case, Mini is a desktop robot. That means that it cannot move in a room. That limits the interaction depending if the user is far or near. However, the robot has many capabilities to attract the attention of people.

In relation to the hardware components included in the robot (Fig. 5.4), from top to bottom we can find first the head. Mini's head includes two screens that work as lively eyes, two illuminated cheeks (RGB) to express some emotions, and a vumeter that shows when it is talking.

Between the head and the torso, a two degrees of freedom (pan and tilt) neck is placed to give movement to the head. Following, we can find the torso with two arms (one degree of freedom). This torso includes a colorful heart to also show emotions

in the outside. Inside of the torso, we can find some touch sensors and a microphone with two speakers in the belly for conversations.

At the bottom, a mobile RGBD camera (kinect) is placed for people detection and identification. The static base of the robot stores the main computer with an arduino microcontroller for data acquisition in addition to a small battery for safe power off.

Externally, a touch screen tablet can be also used to show videos, images, or button menus for interacting in games or exercises. Besides, a push-to-talk button is connected to the robot to activate the user speech recognition in the right moment.

Fig. 5.4 Hardware architecture for the social robot Mini.

Mini robot is a useful platform to interact with people in reception desks of a bank, a museum or even a university covering most HRI needs.

5.3 Experimental Design

In order to design the experiments, we have followed a well-known methodology previously used in other scientific works and, in particular, in many HRI studies (e.g. Breazeal et al. [46], Senft et al. [118], and Senft et al. [119]). For the different experiments implemented, we have followed the next structure:

1. A brief introduction about the experiment is described.

Experimental Results

2. Some **hypotheses** are proposed.
3. The **task** of the experiment is explained to clarify the initial conditions and the usual behavior expected.
4. The **implementation** is explained showing the personality configured for the robot, that is, the drives, motivations, users, states, actions, etc.
5. The **participant** description is presented in addition to their profiles.
6. The **results** obtained are shown together with the evaluation of the hypotheses proposed at the beginning.
7. Finally, the **discussion** of the results is presented.

Through these experiments, we are able to extensively test our system in a real HRI scenario. For that, we evaluate the wellbeing of the robot and the satisfaction of the user. The wellbeing is the reinforcement during the learning phase that the robot is able to control by learning a correct policy. Indirectly, we also evaluate the satisfaction perceived by the robot to promote a social interaction between them.

5.3.1 Scenario description

The socially interactive robot Mini is located in a room where a user is able to approach or move freely near to the robot, being able to interact with the robot by playing games. As we commented before, the scenario for the experiments is restricted to only one user per interaction in order to simplify the perception process.

When the user wants to interact with the robot, he is able to sit down in front of the robot at a distance between 0.5 and 1.5 meters approximately (Figure 5.5). The user will be able to interact through a tablet near to the robot that could be moved to make easier the interaction. The user is able to decide what to do at any time with no restrictions.

5.3.2 Hypotheses

Two hypotheses have been tested with these experiments:

- H1: *Effectiveness and Efficiency of the system*. The robot is able to manage the environment through the states of the users, learning how to maintain its wellbeing as higher as possible.



Fig. 5.5 Positions for the user and the robot when interacting.

- H2: *User's satisfaction*. The robot interacts with the user as many times as possible in order to satiate the need related to the satisfaction of the user.

5.3.3 Task

The tasks developed by the robot try to achieve several objectives. First is related to its needs, where the robot has to try to be satiated as long as possible. The second is related to the satisfaction of the user, where the robot has to try to make the correct actions to achieve the maximum satisfaction of a user. And the third is related to the learning of the robot, the robot has to acquire the correct policies from interactions in the exploration phase.

Depending on the state of the user, the robot will select one of the available actions in the repertoire. Every action will always have an effect over Mini's drives that will make the robot learns about the user. Positive or negative effects of an action will influence the behavior of the user in future interactions.

Although it is the robot who decides what action to make, we have defined some common steps in order to explore all possibilities. Next, we define the usually actions the user and the robot could do:

1. The user is absent from the room. The robot may do nothing or express its mood by attracting the attention.

Experimental Results

2. A user appears in the scenario. The robot may try to attract his attention or do nothing.
3. The user is near to the robot. The robot may keep trying to attract him more closely or do nothing again.
4. The user approaches to the robot. The robot may greet the user and tell him to play a game or be quiet.
5. When interacting, the robot may start playing a game with different difficulty levels or rest. The user is able to answer the question and set a satisfaction score, or leaves the interaction.
6. The user stops the interaction and leaves the interaction area. The robot may try to retain the user to keep the interaction or do nothing.
7. The user moves away from the robot. The robot may try again to attract his attention in order to retain him or keep resting.

5.3.4 Implementation

As we commented in previous sections, each experiment session is composed of a training phase (exploration) and a testing phase (exploitation). The training phase is composed by three individual sessions of around one hour and a half. After the user completed the exploration sessions, one more session is done to evaluate the actions learned.

The motivation to establish these durations is to achieve stable *Q-Values* at the learning process. While the exploration phase, the balance between exploration and exploitation is achieved by modifying the *learning rate* (α). The *temperature* is set to high value (100) to benefit the random selection of the actions independently of the Q-Value associated. On the other hand, the learning rate parameter is set to a high value (0.3) at the beginning, decreasing it 0.1 in the following sessions. As we commented in Section 3.7, the learning rate makes that vary the importance of past learned experiences what benefits the stabilization of the Q-Values. In total, every experiment has a duration of five hours in total. Figure 5.6 graphically resumes the sessions.

We have set also a minimum requisite to keep a well balance experiment. That way, during the exploration phase the user will be around 15 minutes absent, around 15 minutes transitioning between the states absent and near, and around 30 minutes

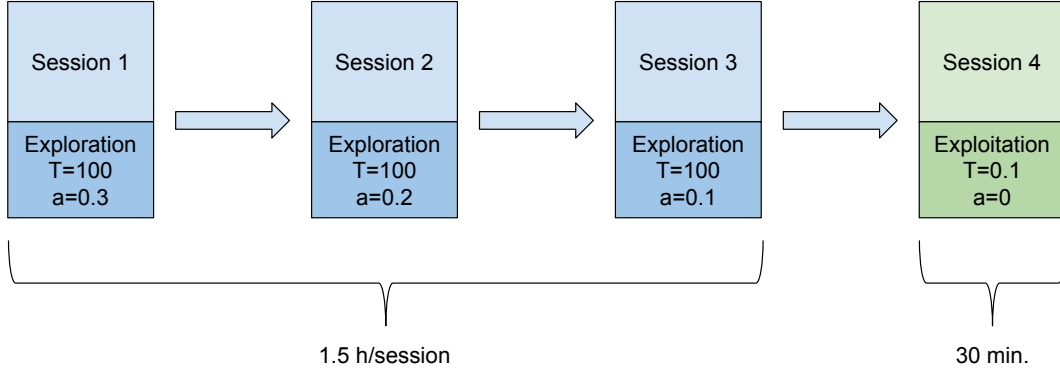


Fig. 5.6 Sessions defined for the HRI experiments.

transitioning between the near and interacting states. Finally, the desire time in the interacting state should round approximately 30 minutes of the total time of an experiment session.

For the set of sessions, we have defined motivations related to the **empathy**, the **relax**, and the **social** needs. Analogously, the **user's satisfaction**, **rest**, and **interaction** drives are related to these motivations. Table 5.1 shows the association between motivations and drives.

Motivation	Drive
empathy	user's satisfaction
relax	rest
social	interaction

Table 5.1 Motivations-Drives relation.

How drives evolve is one of the most important parts of the work related to the decision-making process because they define the **personality** of the robot. For these experiments, **drives** have been designed to evolve differently in each case. However, all drives have an initial value of zero.

The *user's satisfaction* drive (Fig. 5.7) follows a constant function that varies depending on the user input when playing the game. With the aim of determining the satisfaction of a user, an educative game has been developed as a skill. The

Experimental Results

implementation provides a useful way of asking to the user his satisfaction with a button menu in a tablet and in a range from 1 to 3 stars. In this way, we have the answer directly from the opinion of the user and not trying to guess though a perception system. The saturation level for this drive is not limited as well as the satisfaction time, so it can reach the maximum value of 100 that is the upper limit for the saturation.

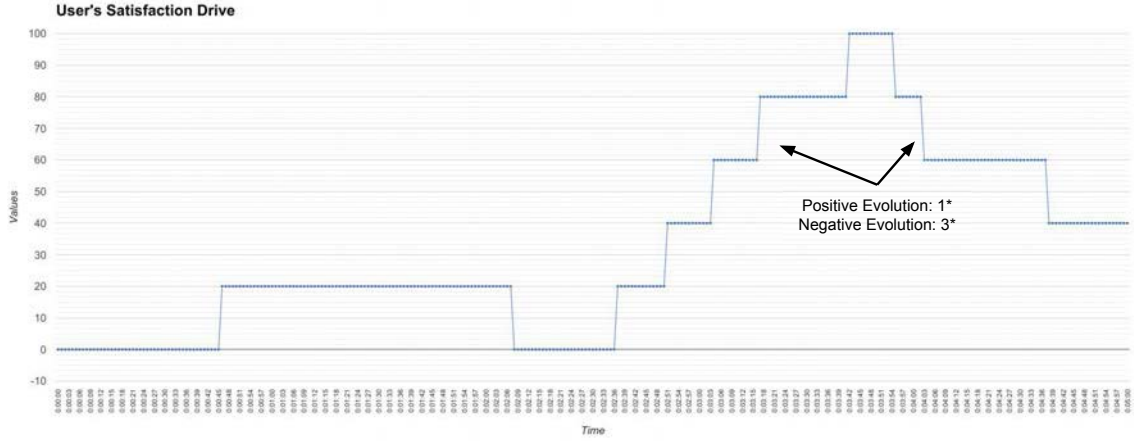


Fig. 5.7 Evolution chart for the user's satisfaction drive.

The *rest* drive (Fig. 5.8) on the contrary follows a linear function that increments sequentially 1.6 points every six seconds and decrement similarly when the robot is executing the action to do *nothing*. This drive has defined a saturation level of 80 with a satisfaction time of two minutes.

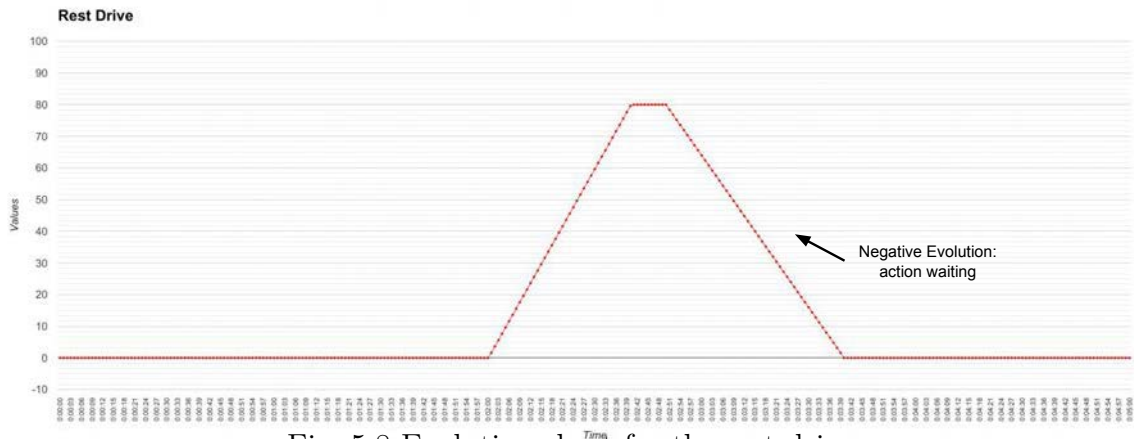


Fig. 5.8 Evolution chart for the rest drive.

On the other hand, the *interaction* drive (Fig. 5.9) evolves also through a linear function that increments 2 points every five seconds when the user is in one of the

states different to the *interaction* state. In that state, the drive will be decreasing its value with the same slope. The saturation level for this drive is established in a value of 90, and a satisfaction time of one minute.

The *interaction* and *relax* drives has been modeled to evolve in the time in order to always have at least one dominant motivation different to the no-motivation.

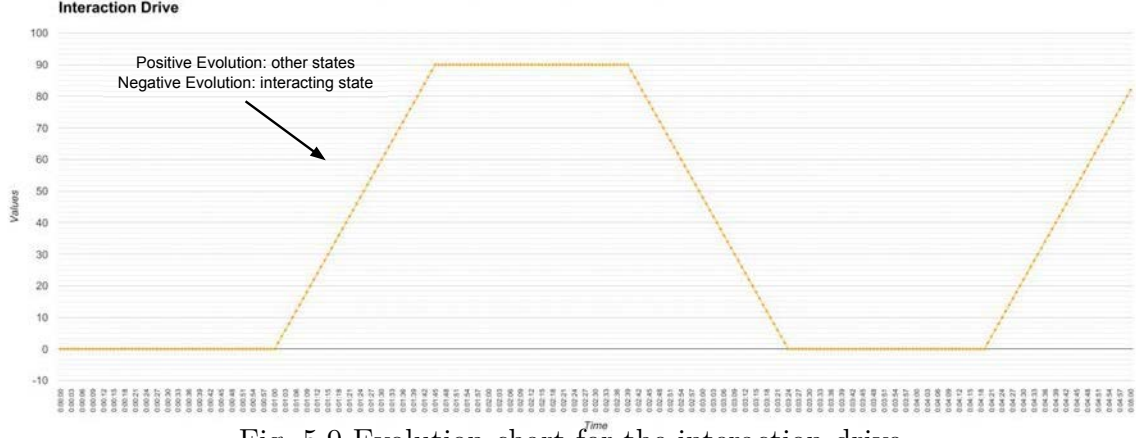


Fig. 5.9 Evolution chart for the interaction drive.

We have also defined the **external stimuli** associated to the motivations. An external stimulus is always related to one state that may affect one or more motivations. For our case, when a user *appears*, the *social* motivation increases its value in 10 points, that is, an increment of the social need. When a user is *near* to the robot, similarly the *social* need increases now its value in 5 points.

As we have commented, more than one motivation may be affected in a state. This is the case when a user is *approaching*. The *relax* and the *social* needs increase their values in 10 points. Finally, when a user begins *interacting*, the *relax* need increases its value in 10 points.

Table 5.2 shows the external stimuli for the experiments.

The repertoire of **actions** available for the robot have been modeled in three skills acting in a different way depending on the input parameter given. All actions are available for each user present in the system modeled as a different Q-Value. This is what allows us to personalize the learned actions for each user. Table 5.3 shows the repertoire of these actions.

The first skill implemented is the **waiting skill**. This skill is made in order to model how to do nothing. The waiting skill will use the *etts skill* to say that is “tired” or similar sentences starting to do nothing the time included as an input parameter.

Experimental Results

State	Motivation	External Stimulus
appearing	social	+10
near	social	+5
approaching	relax	+10
	social	+10
interacting	relax	+10

Table 5.2 Motivations-External stimulus relation.

The waiting skill is for instance very useful for robots with batteries, e.g. Maggie, where it is needed a minimum time for the completion of the charge. But it is also helpful for any robot because when there is nobody near to the robot, not being able to interact. For the experiments, the time set to satiate the need is 5 minutes.

The second skill implemented is a multi-purpose action that **attracts the attention** of a user in different ways. The skill basically uses the *etts skill* to say a sentence in order to bring closer the user for a future interaction. We have defined four distinct inputs to model how the action has to be launched. These parameters are sent depending on the state of the user.

- When the user is *absent*, the attract attention skill will receive that is “alone” saying a sentence such as “Is anybody there?”.
- When the user is present, that is, in the *appearing* or *near* states, the skill will receive the parameter “accompanied” saying something like “Does anyone want to play with me?”.
- When the user is *approaching*, the skill will receive the “greet” parameter and greeting the user.
- When the user leaves the interaction, the skill will receive the “retain” parameter trying to retain the user by saying something like “Sorry, are you leaving?”.

The third skill developed is an **educational game** implemented as math questions that receives the level of difficulty for the question to be answered. The game works as

Action Name	Topic start/stop	Value
nothing	waiting_skill	300
show_question	math_game_skill	easy
show_question	math_game_skill	medium
show_question	math_game_skill	hard
attract_attention	attention_skill	alone
attract_attention	attention_skill	accompanied
attract_attention	attention_skill	greet
attract_attention	attention_skill	retain

Table 5.3 Repertoire of actions available for the DMS.

follows: if it is the first time the user plays the game, an explanation of the rules is given to the user. Following, the question is selected randomly depending on the level of difficulty. The user has some time to answer. After that time, a timeout is launched when no answer is received in order to continue doing other actions. On the other hand, if the user answers the question, the result is provided to the user graphically in the tablet and through speech. Finally, the robot asks the user what level of satisfaction has in that moment in a 1 (low) to 3 (high) “stars” scale. If no answer is received, another timeout is launched, otherwise the satisfaction is sent to the DMS.

All skills implement an interruption manner to easily cancel an action and start the next action selected by the DMS.

We consider that every action has an effect. These effects can be the result of terminating successfully an action or by an interruption decision. Depending on the result of the action that was executing, drives can be modified to model a variation in the needs. Moreover, actions produce a reward that will be used in the learning process.

Table 5.4 shows the relation between actions and their effects.

Experimental Results

State	Action	Effects (over drives)
absent	attract attention “alone”	<i>success</i> : +5 for rest
appearing	attract attention “accompanied”	
near	attract attention “accompanied”	
approaching	attract attention “greet”	
leaving	attract attention “retain”	
interacting	show question “easy”	<i>success</i> : -10 for user’s satisfaction and +5 for rest
	show question “medium”	<i>fail</i> : 0 for user’s satisfaction and +5 for rest
	show question “hard”	<i>error</i> : +10 user’s satisfaction and +5 rest

Table 5.4 Actions-Effects relation.

5.3.5 Participants

With the objective of testing the decision-making system and the learning process, we have evaluated the system with different types of user profiles. These profiles have been defined based on the theory of Bartle [120] with the aim to create real human profiles. Bartle conducted research in the areas of game design and game development, as well as explored player personality types for Multi-User Domain games (MUDS). His theory on game participant psychology classifies players based on their gaming preference.

These preferences are deduced from a series of thirty random questions which identify characteristics associated with specific character types. Bartle identified four main character types – **Achievers**, **Explorers**, **Socializers**, and **Killers**. The results from this test produced a metric known as the “Bartle Quotient” which represent the relative presence of each characteristic trait.

The four main player types are characterized in the following descriptions:

Achievers Players who focus on obtaining some level of success, as measured by points, prizes, material possessions, or other valuation criteria. Known as the

”Diamonds“, they will strive to gain rewards, recognition and prestige, with little or no advantage in gameplay or advancement.

Explorers Players who seek out the thrill of discovery, learning about anything that is new or unknown. Referred to as the ”Spades“ because they tend to dig down and uncover things, explorers feel a rush of excitement when they discover a rare artifact or a secret pathway.

Killers These players live for the competitive elements of the game. They are referred to as the ”Clubs“ because they like to “take it to” their competition. They love the opportunity to compete (and beat) the other players.

Socializers These are individuals who are attracted to the social aspects of a game, rather than the game strategy itself. They are the ”Hearts“ of the game world, because they gain the most enjoyment from interacting with the other players in the game. For them the game is the social vehicle that allows them to engage others and build interesting relationships.

Based on these profiles, we have adapted them for our HRI experiments translating them in two types of roles.

- **User Profile 1.** We combined the *achiever* and the *explorer* profiles to get a kind of user that behaves in two ways. When not interacting, the user is attracted the first times when the robot act but, once he knows the robot, he is going to interact only when he wants. On the other hand, when interacting, the user behaves in the next manner: easy questions increase his satisfaction, medium level questions do not affect to his satisfaction, and difficult questions decrease his satisfaction.
- **User Profile 2.** For this kind of user, we combined the *killer* and *socializer* profiles in order to have an antagonistic profile. In this case, when not interacting, the user has the necessity to be social that is translated to an approach when the robot tries to attract the attention of users. On the other hand, when interacting, this user behaves in the next manner: easy questions decrease his satisfaction, medium level questions do not affect to his satisfaction, and difficult questions increase his satisfaction.

5.4 Results

In this section we present the results for the experiments developed from the user profiles defined previously.

5.4.1 Effectiveness and Efficiency of the system

In order to measure the effectiveness and efficiency of the system we present the results of the actions learned for every type of user form the exploration phase. Later, the wellbeing of the robot in the exploitation phase is analyzed.

5.4.1.1 Learning by doing

The experiments are associated to a each type of user profile that has its own behavioral conditions. The process of learning defines how to act but usually is a long time process that needs a lot of interactions. With the experimental design we have made, we are able to check the actions learned after the exploration phase.

Firstly, Table 5.5 presents the results for the actions established by the robot as the best option when the *user profile 1* is present.

State/Motivation	User's satisfaction	Relax	Interaction
absent	nothing	nothing	nothing
appearing	nothing	nothing	nothing
near	nothing	nothing	nothing
approaching	nothing	nothing	nothing
interacting	show question easy	show question easy	show question easy
leaving	nothing	nothing	nothing
disappearing	attract attention	nothing	nothing

Table 5.5 Actions learned (maximum Q-Value) for the user profile 1.

Just remember before the personality for the *user profile 1*. This user behaves in this way:

- When not interacting, the user is attracted the first times when the robot act but, once he knows the robot, he is going to interact only when he wants.
- When interacting, the user behaves in the next manner:
 - Easy questions increase his satisfaction.
 - Medium level questions do not affect to his satisfaction.
 - Difficult questions decrease his satisfaction.

Remember also that the effect of the execution of the action “attract attention” produces an increment in the *rest* drive. On the other hand, the “nothing” action produces an effect decrementing the *rest* drive depending on the time it is executing.

Although we can check that the personality of the robot is not the most social in this case, the robot has learned correctly what are the actions that increase more its wellbeing to the extent possible. Likewise, observing when the user is in the *interacting* state, we can observe that the robot has also learned correctly that when making easy questions the satisfaction of the user is increased as well as its wellbeing.

Exceptionally, there is a situation where the action does not correspond with the profile of the robot. This is when the *user’s satisfaction* motivation is dominant and the state of the user is *disappearing*. In this case, it is due to a low exploration of the Q-Value. In Q-Learning, when a pair $\langle state, action \rangle$ is not explored a minimum number of times, the resultant action may not correspond to the real result because it has not had enough time to converge.

Secondly, Table 5.6 shows the results for the actions learned by the robot when the *user profile 2* is in the environment.

Consider before the personality for the *user profile 2*. Remember that this user behaves in this way:

- When not interacting, the user has the necessity to be social that is translated to an approach when the robot tries to attract the attention of users.
- When interacting, the user behaves in the next manner:
 - Easy questions decrease his satisfaction.
 - Medium level questions do not affect to his satisfaction.

Experimental Results

State/Motivation	User's satisfaction	Relax	Interaction
absent	attract attention	nothing	nothing
appearing	nothing	nothing	nothing
near	nothing	nothing	nothing
approaching	attract attention	nothing	nothing
interacting	show question hard	show question hard	show question hard
leaving	attract attention	nothing	nothing
disappearing	attract attention	nothing	nothing

Table 5.6 Actions learned (maximum Q-Value) for the user profile 2.

- Difficult questions increase his satisfaction.

Focusing initially when the user is in the *interacting* state, the robot has correctly learned that playing hard questions increase both the satisfaction of the user and its wellbeing.

We have also in this experiment a situation where the results are not totally reliable. This is the case when the dominant motivation is the *user's satisfaction* and the *absent* and *disappearing* states. The low number of times these values have been explored makes the results inaccurate.

Nevertheless, for the *relax* and *interaction* motivation, the results fit perfectly with the personality of the robot because of the effects of the actions. Particularly, due to the backpropagation reward that occurs during learning, the robot has learned that the best way to satiate this motivation is executing the *attract attention* action.

5.4.1.2 Wellbeing

We also include another measure to evaluate if the robot has learned correctly from the exploration phase, that is the wellbeing of the robot. The wellbeing during that phase is not a reliable metric because the actions are randomly executed. This is done in order to explore all possible $\langle \text{states}, \text{actions} \rangle$ pairs. In this way, the last session (exploitation) of every experiment gives us more reliable results.

In the exploitation phase each user profile has followed the same behavior as defined previously. Additionally, the robot starts with the need of satisfying the user as dominant in order to promote the interaction. During the experiment time, the participant will interact with the robot depending on the features of the profile. However, for the *user profile 1* we suppose that he will interact at least two times. For both cases, the participant starts in the *absent* state appearing after a time.

Figure 5.10 shows the evolution of the wellbeing of the robot during the exploitation phase when the *user profile 1* is present.

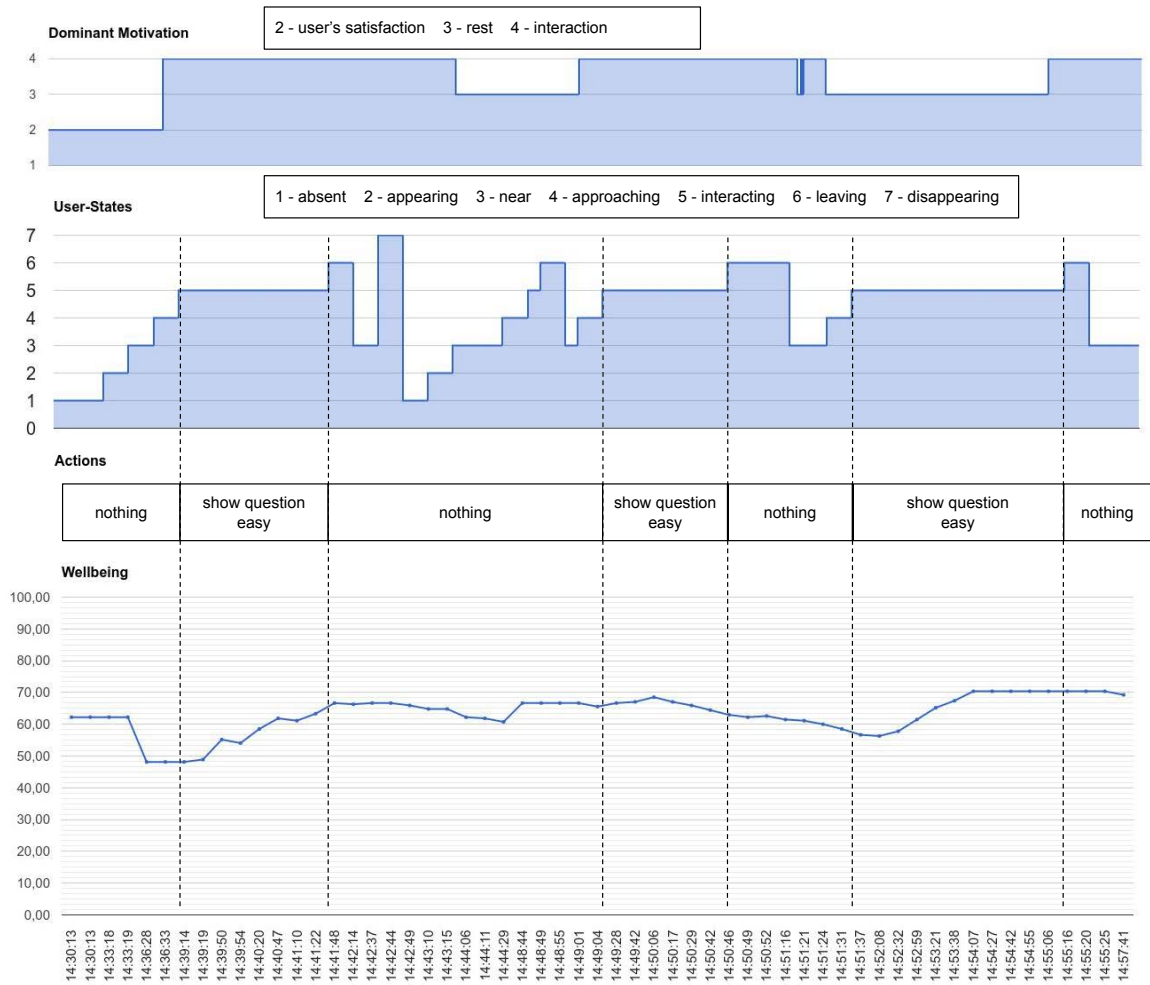


Fig. 5.10 Wellbeing of the robot in the experiment with user profile 1.

In this experiment, we can observe that the wellbeing initially starts decreasing whilst the user is *absent* and until the saturation of the *interaction* drive. When the user appears, the robot continues doing *nothing* until the user moves closer with the objective of interacting. In that moment, both begin to play the game until the user

Experimental Results

leaves. Here we can notice an increment in the wellbeing because both the *user's satisfaction* and the *interaction* needs are being satiated at the same time. Later, when the user leaves the interaction, the robot starts doing *nothing* again. This action provokes a satiation on the *rest* drive but the *interaction* need is increasing on the contrary. The same pattern is repeated two more times with the same effect in the wellbeing of the robot but now with the particularity that the *user's satisfaction* need has been satiated previously.

Figure 5.11 shows, on the contrary, the evolution of the wellbeing of the robot during the exploitation phase when the *user profile 2* is present.

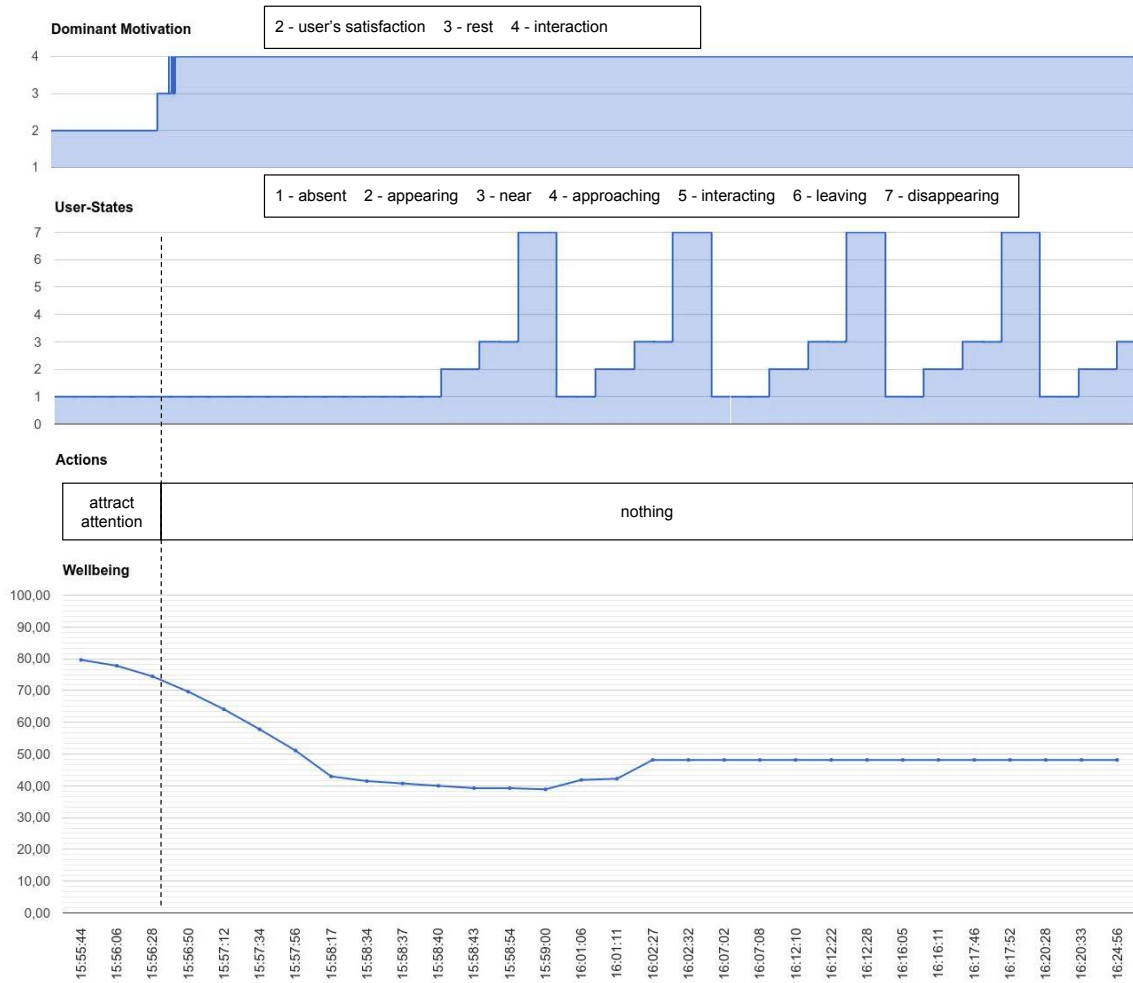


Fig. 5.11 Wellbeing of the robot in the experiment with user profile 2.

For the second experiment, we can notice that, analogously to the first experiment, the wellbeing of the robot starts decreasing while the user is *absent* stabilizing when the *interaction* drive is saturated. During that process, the robot has been executing

the *attract attention* action with no effect. When the dominant motivation changes, the robot starts doing *nothing* continuously. This behavior is because the user profile of this experiment only interacts when the robot tries to attract his attention. Although the user moves closer to the robot, this does not emit any intention to attract his attention so the interaction never happens. Even so, we can observe that the robot has learned a not very social policy but that stabilizes the wellbeing.

Summarizing, Table 5.7 shows a comparative of the average wellbeing for both kind of users evaluated. We can observe that the *user profile 1* is the user with whom the robot is most satiated in average. Even so, the robot is able to achieve in both experiments a wellbeing above the middle of the table.

	UP1	UP2
AVG	63.22	50.51
STD	5.92	11.14

Table 5.7 Average and standard deviation for the wellbeing of the robot.

5.4.2 User’s satisfaction

This metric intends to evaluate how satisfied has been an user during the interaction. Remember that the satisfaction is measured directly from the interactions. Every time a user interacts with the robot playing the educational game, at the end of the game, the robot asks the user how satisfied is in a scale from 1 to 3 “stars”. Thus, as explained in Section 5.3.4, the game will send a *success* when 3 stars, a *fail* when 2 stars, and an *error* when 1 star.

At the beginning, the *user’s satisfaction* need starts with an initial value of 50. Table 5.8 shows the percentage of time the robot has perceived the users were satisfied during the exploitation phase.

UP1	UP2
60%	0%

Table 5.8 Percentage of time the robot has perceived the user is satisfied.

Experimental Results

We can observe for the first experiment that the robot has learned the actions to achieve that a 60% of the time the user is satisfied. On the contrary, with the actions learned for the second experiment, the robot is not able to attract the user not being possible the interaction. This produces that the satisfaction of the user never changes.

5.5 Discussion

Since the configuration of the personality of the robot has a high level of influence in the decision-making system. Depending on how the DMS is configured, the actions learned and the evolution of the drives define the sociability of the robot. Therefore, there is many open challenges in HRI that could be studied with autonomous Socially Interactive Robots.

The results we have obtained are then conditioned to the personality we have defined for the robot. Where the need of interaction is quickly incremented in order to favor the socialization. Besides, the need of relax is designed to be inversely proportional to the need of interaction. Thus, the robot has always a dominant motivation that makes it more lively.

With these results we have proved that the system is working properly. Nevertheless, the learning method used is still conditioned to the time of exploration. The Q-Learning methodology requires a large time of exploration that we have reduces considerably through interaction. However, for long-term interactions, our experiment is still short due to the low exploration of some the Q-Values. The exploration phase should then be a little longer in order to acquire reliable results in the exploitation phase.

We have also checked that the wellbeing of the robot is very similar from one user profile to the other. We have achieved that in both experiments the robot has an average wellbeing above 50.

Finally, we have analyzed if the robot is able to complete the satisfaction of the users. Observing that, with the personality set, the robot is only capable of satisfy one of the user's profile defined.

Even so, this results are only the first step in the research of autonomous decision-making in SIR. Where long-term interactions in spaces with robots and humans is the common scenario to achieve.

5.6 Conclusion

We have presented the robots that we used during the implementation and the evaluation of the system. Besides, we presented a experimental scenario in which we described the methodology we used in order to evaluate the complete system in a real environment. Thus, we defined the scenario together with the tasks the robot and the participants did. Additionally, we defined two general profiles for the participants based on theory of games. The results obtained has validated our system through an extensive experimentation organized in different sessions.

Those results validated the hypotheses addressed confirming that the system is working properly when the robot is learning from interactions. Moreover, the wellbeing of the robot has been analyzed in order to check that the exploration phase was well defined.

Finally, we discussed the benefits and shortages of the system giving some previous comments to the future improvements.

Chapter 6

Conclusions and Future Works

I'm as proud of many of the things we haven't done as the things we have done. Innovation is saying no to a thousand things.

Steve Jobs

6.1 Conclusions

This thesis has contributed to the fields of autonomous decision-making and robotic architectures by combining techniques from the fields of Software Engineering, Human-Robot Interaction, and Machine Learning, among others. This chapter concludes this thesis by summarizing the main contributions related to these topics in addition to present the future works that remain as open challenges in these research topics.

The contributions of this thesis enabled Socially Interactive Robots to make decisions in an autonomous manner learning from the interactions with people. We tested our system in a real scenario where the robot is continuous learning in order to maximize its wellbeing. Additionally, the robot learns the actions to try to improve the satisfaction of the user present in the interaction.

A major contribution of this thesis has been improving the autonomous capabilities of Socially Interactive Robots with the inclusion of a bio-inspired decision-making system for human-robot interaction. The aim of this was to enable the robot to learn better and faster from the interactions it made with the different types of users studied during the exploration sessions.

6.2 Summary of the Key Contributions

The main contributions of this thesis are summarized as follows:

Regarding decision-making systems: We developed a system that endows a robot to make decisions while interacting with people according to the user's behavior. For such system we developed four approaches:

- The first one focused designing a bio-inspired decision-making system that is centered in human-robot interaction for real applications.
- The second approach consisted in monitoring the states of the robot in real time. Adding more control to the DMS and evaluating the decisions when real changes in the environment are occurring.
- We managed exogenous actions, actions made by others, modeled as the states related to the users. This allows the robot to have a more controlled environment by actions that does not depend on it.
- We proposed a method that overcomes the problem of interrupting actions in real applications. This method provides to the system a generalized way for acting in any interaction with people.

Regarding robotic architectures: We developed a robotic architecture from low- to high-levels of abstraction. Here, we have contributed in two different manners:

- We have developed a software architecture based on software methodologies that provides several benefits for SIR such as easy of reuse, modularity, and portability to any robot.
- We provided an implementation of the architecture based on three layers adding novelty concepts from cognitive science such as memories and skills. For that, we used the ROS framework that provides useful libraries for action execution and interruption.

Regarding human-robot interaction: We provided an autonomous decision-making system developed in a robotic architecture deployed in real robots for interacting with different types of users. For such field we contributed in different ways:

- We evaluated the system in a scenario where the balance between exploration and exploitation is the main objective in order to provide the enough knowledge to the robot.

- We adapted a well-known study of types of users redefining them to HRI.
- We obtained some relevant results that help us to understand how the personalities of the robots has to be modeled in order to be more social.

6.3 Future Works

Decision-Making Our work leaves other paths open for exploration:

- First, the work developed in this thesis opens the door for building autonomous robots where the decision-making process can be included in order to a better interaction with people. Commercial robots can now improve their autonomy including a system like proposed in this thesis.
- Second, other bio-inspired and cognitive concepts such as the semantic memories or emotions can be included. The relevant role of emotions have not been discussed in this dissertation, having a real influence in the bio-inspired decision-making processes both for the robot and the users.
- Third, with only a few minor modifications, the DMS can be extended with other learning methodologies. Either by including non-supervised learning techniques or by applying additional bonuses to the reward of the actions the system would include techniques such as teaching robotics.

Repertoire of Skills Regarding our repertoire of skills, several future works are still open:

- From the HRI point of view, the system remains open for future skills that acquire more information about the users such as detecting when the user is tired or bored when playing a game, or in a conversation to obtain personal information that later can be used in a personalized interaction. This provides the robot a more complete profile of them. Besides, this feature will help the skills and the DMS to provide a more natural interaction. It is also related to this point the interaction with more than one person at the same time. When systems of perception are able to give more information to the high-level layers.
- From the application point of view, increasing the number of actions (repertoire of skills) will provide to the system more possibilities to experiment. Other

Conclusions and Future Works

applications such as entertaining or cognitive rehabilitation are highly related being easily to include as new skills.

- From the educational point of view, the game implemented in this thesis comes from the educational field. It might be interesting to study other educational games improving the interaction with children in schools in order to help increase interest in technology.

Experiments in HRI Most of the future works for the Socially Interactive Robots are related to the human-robot interaction field. The most interesting in this area can be:

- Showing the results of this thesis, new open researches can be done in the field of HRI. Next step in the process can be testing other motivations in order to provide other personalities to the robot.
- To apply the DMS to other areas such as medicine or entertainment. Accompanying children or elder in hospitals, or just for fun in homes.
- Due to the limited repertoire of actions, autonomous robots have a different impact on people who expects human-like functionalities when not implemented. Several studies can be made in this field evaluating the expectations of the users with robots with different repertoires of possible actions.

6.4 List of Publications

Preliminary results of this thesis have already been published.

6.4.1 Journals

Journals published are indexed in the Journal Citation Reports (JCR).

- **R. Pérula-Martínez**, J. M. García-Haro, C. Balaguer, and M. A. Salichs. “Developing Educational Printable Robots to Motivate University Students Using Open Source Technologies”. *Journal of Intelligent and Robotics Systems* (Q3). 2015.

6.4.2 Conferences and Workshops

- **R. Pérula-Martínez**, Álvaro Castro-González, María Malfaz, Miguel A. Salichs. “Autonomy in Human-Robot Interaction Scenarios for Entertainment”. 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI). 2016.
- F. R. Cañadillas, **R. Pérula-Martínez**, V. González, M. A. Salichs, and C. Balaguer. “ProtoCREA: A Robot to Teach Them All”. EDULEARN16. 2016.
- V. González, F. R. Cañadillas, **R. Pérula-Martínez**, M. A. Salichs, and C. Balaguer. “A Review on How to Easily Program Robots at High School”. RoboCity16. Robots for Citizens. 2016.
- V. González, **R. Pérula-Martínez**, F. R. Cañadillas, M. A. Salichs, and C. Balaguer. “Estado de la Tecnología en Robótica Educativa para la Educación Secundaria”. XXXVI Jornadas de Automática. 2015.
- **R. Pérula-Martínez**, E. Salichs, I. P. Encinar, Á. Castro-González, and M. A. Salichs. “Improving the expressiveness of a social robot through luminous devices”. 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI). 2015.
- **R. Pérula-Martínez**, A. Al-Kaff, and J. M. García-Haro. “Diseño de un robot móvil como mascota robótica de entretenimiento para personas con discapacidad”. 13th workshop Robocity2030. Robots para los ciudadanos. 2013.
- **R. Pérula-Martínez**, J. M. García-Haro, and A. Al-Kaff. “Modelado e Implementación de un robot de entretenimiento para competición”. 13th workshop Robocity2030. Robots para los ciudadanos. 2013.

6.4.3 Source Code

In addition to text publications, we released some of the components developed for our SIR in the official [GitHub account](#) of the Social Robots Group (RoboticsLab) from the UC3M with an Open Source license. These are the related to the drives and devices of the robots.

Additionally, we have included in the ROS [documentation](#) with a complete description of the robotic architecture of the socially interactive robot Maggie. Contributing the community with the main components the architecture has

Conclusions and Future Works

implemented and the link to the source code developed. Including all components from drivers, devices and high-level skills developed in the architecture.

References

- [1] United Nations Population Division, “World Population August 2016.” [Online. Last accessed: 2016-08].
(Cited on page 1)
- [2] National Science Foundation (NSF), “National Robotics Initiative invests 38 million dollars in next-generation robotics.” [Online. Last accessed: 2016-08].
(Cited on page 2)
- [3] T. Fong, I. Nourbakhsh, and K. Dautenhahn, “A survey of socially interactive robots,” *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 143–166, 2003.
(Cited on page 2)
- [4] C. Bartneck and J. Forlizzi, “A design-centred framework for social human-robot interaction,” in *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No.04TH8759)*, pp. 591–594, IEEE, 2004.
(Cited on page 3)
- [5] B. R. Duffy, C. Rooney, G. M. P. O’Hare, and R. O’Donoghue, “What is a Social Robot?,” in *10th Irish Conference on Artificial Intelligence & Cognitive Science*, pp. 1–8, University College Cork, 1999.
(Cited on page 3)
- [6] D. Feil-Seifer and M. Mataric, “Socially Assistive Robotics,” in *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, pp. 465–468, IEEE, 2005.
(Cited on page 3)
- [7] B. R. Duffy, “Fundamental Issues in Social Robotics,” *International Review of Information Ethics (IRIE)*, vol. 6, no. 12, pp. 31–36, 2006.
(Cited on pages 3 and 19)
- [8] F. Hegel, C. Muhl, B. Wrede, M. Hielscher-Fastabend, and G. Sagerer, “Understanding Social Robots,” in *2009 Second International Conferences on Advances in Computer-Human Interactions*, pp. 169–174, IEEE, 2009.
(Cited on page 3)
- [9] B. Chandrasekaran and J. M. Conrad, “Human-robot collaboration: A survey,” in *SoutheastCon 2015*, pp. 1–8, IEEE, 2015.
(Cited on page 3)

References

- [10] G. Hoffman and C. Breazeal, “Collaboration in Human-Robot Teams,” in *AIAA 1st Intelligent Systems Technical Conference*, (Reston, Virigina), pp. 1–18, American Institute of Aeronautics and Astronautics, 2004.
(Cited on page [3](#))
- [11] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, “Towards robotic assistants in nursing homes: Challenges and results,” *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 271–281, 2003.
(Cited on page [3](#))
- [12] A. Causo, G. T. Vo, I.-M. Chen, and S. H. Yeo, “Design of Robots Used as Education Companion and Tutor,” in *Mechanisms and Machine Science*, vol. 37, pp. 75–84, Springer, 2016.
(Cited on page [3](#))
- [13] K. Stubbs, D. Bernstein, K. Crowley, and I. Nourbakhsh, “Long-Term Human-Robot Interaction: The Personal Exploration Rover and Museum Docents,” in *Proceeding of the 2005 conference on artificial intelligence in education*, pp. 621–628, IOS Press, 2005.
(Cited on page [3](#))
- [14] M. Shiomi, T. Kanda, I. Howley, K. Hayashi, and N. Hagita, “Can a Social Robot Stimulate Science Curiosity in Classrooms?,” *International Journal of Social Robotics*, vol. 7, no. 5, pp. 641–652, 2015.
(Cited on page [3](#))
- [15] C. Plaisant, A. Druin, C. Lathan, K. Dakhane, K. Edwards, J. M. Vice, and J. Montemayor, “A storytelling robot for pediatric rehabilitation,” in *Proceedings of the fourth international ACM conference on Assistive technologies - Assets '00*, pp. 50–55, ACM Press, 2000.
(Cited on page [3](#))
- [16] A. Ferrein and G. Steinbauer, “20 Years of RoboCup,” *KI - Künstliche Intelligenz*, aug 2016.
(Cited on page [3](#))
- [17] F. Tanaka and H. Suzuki, “Dance interaction with QRIO: a case study for non-boring interaction by using an entrainment ensemble model,” in *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No.04TH8759)*, pp. 419–424, IEEE, 2004.
(Cited on page [3](#))
- [18] T. Fong, I. Nourbakhsh, and K. Dautenhahn, “A survey of socially interactive robots: Concepts, Design, and Applications,” *Robotics and Autonomous Systems*, vol. 42, pp. 143–166, mar 2003.
(Cited on page [3](#))
- [19] K. Parisot and A. Flachot, “Social Robotics and Technology Acceptation,” *Master Recherche IC2A Ingénierie de la Cognition, de la Création et des Apprentissages*, pp. 71–80, 2016.
(Cited on page [3](#))

-
- [20] T. Kanda, R. Sato, N. Saiwaki, and H. Ishiguro, “A Two-Month Field Trial in an Elementary School for Long-Term Human-Robot Interaction,” *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 962–971, 2007.
(Cited on page 4)
- [21] L. K. Kheng, D. S. Syrdal, M. L. Walters, and K. Dautenhahn, “Living with robots: Investigating the habituation effect in participants’ preferences during a longitudinal human-robot interaction study,” in *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pp. 564–569, 2007.
(Cited on page 4)
- [22] C. Kidd and C. Breazeal, “Robots at home: Understanding long-term human-robot interaction,” in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3230–3235, IEEE, 2008.
(Cited on page 4)
- [23] I. Leite, C. Martinho, and A. Paiva, “Social Robots for Long-Term Interaction: A Survey,” *International Journal of Social Robotics*, vol. 5, no. 2, pp. 291–308, 2013.
(Cited on page 4)
- [24] J. Forlizzi and C. DiSalvo, “Service robots in the domestic environment,” in *Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction - HRI '06*, p. 258, ACM Press, 2006.
(Cited on page 4)
- [25] J. Sung, H. I. Christensen, and R. E. Grinter, “Robots in the wild: Understanding long-term use,” *Human-Robot Interaction (HRI), 2009 4th ACM/IEEE International Conference on*, pp. 45–52, 2009.
(Cited on page 5)
- [26] Y. Farnaeus, M. Håkansson, M. Jacobsson, and S. Ljungblad, “How do you play with a robotic toy animal?,” in *Proceedings of the 9th International Conference on Interaction Design and Children. ACM*, pp. 39–48, 2010.
(Cited on page 5)
- [27] H. Hüttenrauch and K. Severinson Eklundh, “Fetch-and-carry with CERO: Observations from a long-term user study with a service robot,” *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pp. 158–163, 2002.
(Cited on page 5)
- [28] A. M. Sabelli and T. Kanda, “Robovie as a Mascot: A Qualitative Study for Long-Term Presence of Robots in a Shopping Mall,” *International Journal of Social Robotics*, 2015.
(Cited on page 5)
- [29] D. O. Johnson, R. H. Cuijpers, J. F. Juola, E. Torta, M. Simonov, A. Frisiello, M. Bazzani, W. Yan, C. Weber, S. Wermter, N. Meins, J. Oberzaucher, P. Panek, G. Edelmayer, P. Mayer, and C. Beck, “Socially Assistive Robots: A Comprehensive Approach to Extending Independent Living,” *International*

References

- Journal of Social Robotics*, 2013.
(Cited on page 5)
- [30] M. Lohse, F. Hegel, and B. Wrede, “Domestic applications for social robots: an online survey on the influence of appearance and capabilities,” *Journal of Physical Agents*, vol. 2, no. 2, pp. 21–32, 2008.
(Cited on page 5)
- [31] K. Severinson-Eklundh, A. Green, and H. Hüttenrauch, “Social and collaborative aspects of interaction with a service robot,” *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 223–234, 2003.
(Cited on page 5)
- [32] A. M. Sabelli, T. Kanda, and N. Hagita, “A conversational robot in an elderly care center: An ethnographic study,” *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*, pp. 37–44, 2011.
(Cited on page 5)
- [33] A. Burton, “Dolphins, dogs, and robot seals for the treatment of neurological disease,” *The Lancet Neurology*, vol. 12, no. 9, pp. 851–852, 2013.
(Cited on page 5)
- [34] H. Kozima, M. P. Michalowski, and C. Nakagawa, “Keepon: A playful robot for research, therapy, and entertainment,” *International Journal of Social Robotics*, vol. 1, no. 1, pp. 3–18, 2009.
(Cited on page 5)
- [35] D. Mazzei, N. Lazzeri, L. Billeci, R. Igliozi, A. Mancini, A. Ahluwalia, F. Muratori, and D. De Rossi, “Development and evaluation of a social robot platform for therapy in autism,” in *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*, vol. 2011, pp. 4515–8, 2011.
(Cited on page 5)
- [36] J. A. Mann, B. A. MacDonald, I.-H. Kuo, X. Li, and E. Broadbent, “People respond better to robots than computer tablets delivering healthcare instructions,” *Computers in Human Behavior*, vol. 43, pp. 112–117, 2015.
(Cited on page 5)
- [37] H. Broekens, Joost; Heerink, Marcel; Rosendal, “Assistive social robots in elderly care: a review,” *Gerontechnology*, vol. 8, no. 2, pp. 94–103, 2009.
(Cited on page 5)
- [38] T. Klammer and S. B. Allouch, “Zoomorphic robots used by elderly people at home,” in *Proc. of the 27th Int. Conf. on Human Factors in Computing Systems*, pp. 1–2, ACM Press, 2010.
(Cited on page 5)

-
- [39] T. Klammer and S. B. Allouch, “Acceptance and use of a social robot by elderly users in a domestic environment,” in *Proceedings of the 4th International ICST Conference on Pervasive Computing Technologies for Healthcare*, pp. 1–8, IEEE, 2010.
(Cited on page 5)
- [40] S. Hutson, S. L. Lim, P. J. Bentley, N. Bianchi-Berthouze, and A. Bowling, “Investigating the suitability of social robots for the wellbeing of the elderly,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6974 LNCS, no. PART 1, pp. 578–587, 2011.
(Cited on page 5)
- [41] A. M. von der Pütten, N. C. Krämer, and S. C. Eimler, “Living with a robot companion - Empirical Study on the Interaction with an Artificial Health Advisor,” *Proceedings of the 13th international conference on multimodal interfaces - ICMI '11*, p. 327, 2011.
(Cited on page 5)
- [42] A. Ramachandran, A. Litoiu, and B. Scassellati, “Shaping Productive Help-Seeking Behavior During Robot-Child Tutoring Interactions,” in *2016 ACM/IEEE International Conference on Human-Robot Interaction (HRI 2016)*, 2016.
(Cited on page 5)
- [43] N. Campbell and J. Reece, *Biology*. Pearson Benjamin Cummings, 2008.
(Cited on page 11)
- [44] P. Maes, “A bottom-up mechanism for behaviour selection in an artificial creature,” in *From animals to animats: proceedings of the first international conference on simulation of adaptive behavior*, pp. 478–485, MIT Press, 1991.
(Cited on page 12)
- [45] M. Proetzsch, T. Luksch, and K. Berns, “Development of complex robotic systems using the behavior-based control architecture iB2C,” *Robotics and Autonomous Systems*, vol. 58, no. 1, pp. 46–67, 2010.
(Cited on page 12)
- [46] A. L. Thomaz and C. Breazeal, “Teachable robots: Understanding human teaching behavior to build more effective robot learners,” *Artificial Intelligence*, vol. 172, pp. 716–737, apr 2008.
(Cited on pages 12 and 83)
- [47] M. P. Georgeff and F. F. Ingrand, “Decision-Making in an Embedded Reasoning System,” *Proceedings of the 11th international joint conference on Artificial intelligence*, vol. 2, pp. 972–978, 1989.
(Cited on page 12)
- [48] R. Brooks, “Intelligence without representation,” *Artificial intelligence*, vol. 47, pp. 139–159, 1991.
(Cited on page 12)

References

- [49] D. A. Norman, A. Ortony, and D. M. Russell, “Affect and machine design: Lessons for the development of autonomous machines,” *IBM Systems Journal*, vol. 42, no. 1, pp. 38–44, 2003.
(Cited on page [13](#))
- [50] A. Ortony, D. A. Norman, and W. Revelle, “Affect and Proto-Affect in Effective Functioning,” in *Who Needs Emotions?*, pp. 173–202, Oxford University Press, apr 2005.
(Cited on page [13](#))
- [51] A. Sloman, M. Scheutz, and B. Logan, “Evolvable Architectures For Human-Like Minds,” *Affective minds*, pp. 169–181, 2000.
(Cited on page [13](#))
- [52] F. Ingrand and M. Ghallab, “Deliberation for autonomous robots: A survey,” *Artificial Intelligence*, vol. 1, pp. 1–35, 2014.
(Cited on page [13](#))
- [53] K. C. Berridge, “Motivation concepts in behavioral neuroscience,” *Physiology and Behavior*, vol. 81, no. 2, pp. 179–209, 2004.
(Cited on page [13](#))
- [54] W. Cannon, *The wisdom of the body*. W. W. Norton and Company, 1932.
(Cited on page [13](#))
- [55] J. D. Velásquez, “Modeling Emotions and Other Motivations in Synthetic Agents,” *Fourteenth National Conference on Artificial Intelligence*, p. 10, 1997.
(Cited on page [14](#))
- [56] R. C. Arkin, M. Fujita, T. Tagaki, and R. Hasegawa, “An Ethological and Emotional Basis for Human- Robot Interaction,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, vol. 42, pp. 191–201, 2002.
(Cited on page [14](#))
- [57] D. Cañamero, “Designing Emotions for Activity Selection,” *Emotions in humans and artifacts*, pp. 115–148, 2003.
(Cited on page [15](#))
- [58] D. Cañamero, “Modeling motivations and emotions as a basis for intelligent behavior,” in *Proceedings of the first international conference on Autonomous agents - AGENTS '97*, no. Abbott 1884, (New York, New York, USA), pp. 148–155, ACM Press, 1997.
(Cited on page [15](#))
- [59] D. Canamero, “A hormonal model of emotions for behavior control,” *VUB AI-Lab Memo*, vol. 2006, pp. 1–10, 1997.
(Cited on page [15](#))
- [60] C. L. Breazeal, *Designing sociable robots*. MIT Press, 2004.
(Cited on page [16](#))

-
- [61] V. Vouloutsis, S. Lallée, and P. F. M. J. Verschure, “Modulating behaviors using allostatic control,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8064 LNAI, pp. 287–298, 2013.
(Cited on page 16)
 - [62] H.-L. Cao, P. Gómez Esteban, D. B. Albert, R. Simut, G. Van de Perre, D. Lefebvre, and B. Vanderborght, “A Collaborative Homeostatic-Based Behavior Controller for Social Robots in Human–Robot Interaction Experiments,” *International Journal of Social Robotics*, apr 2017.
(Cited on page 17)
 - [63] Z. Kowalczyk and M. Czubenko, “Intelligent decision-making system for autonomous robots,” *International Journal of Applied Mathematics and Computer Science*, vol. 21, no. 4, pp. 671–684, 2011.
(Cited on page 18)
 - [64] Á. Castro-González, M. Malfaz, and M. A. Salichs, “Learning the Selection of Actions for an Autonomous Social Robot by Reinforcement Learning Based on Motivations,” *International Journal of Social Robotics*, vol. 3, no. 4, pp. 427–441, 2011.
(Cited on page 18)
 - [65] R. C. Arkin, P. Ulam, and A. R. Wagner, “Moral decision making in autonomous systems: Enforcement, moral emotions, dignity, trust, and deception,” *Proceedings of the IEEE*, vol. 100, no. 3, pp. 571–589, 2012.
(Cited on page 18)
 - [66] G. A. Hollinger, Y. Georgiev, A. Manfredi, B. A. Maxwell, Z. A. Pezzementi, and B. Mitchell, “Design of a social mobile robot using emotion-based decision mechanisms,” in *IEEE International Conference on Intelligent Robots and Systems*, pp. 3093–3098, IEEE, oct 2006.
(Cited on page 18)
 - [67] M. Malfaz, *Decision Making System for Autonomous Social Agents Based on Emotions and Self-learning*. PhD thesis, University Carlos III of Madrid, 2007.
(Cited on page 18)
 - [68] H. Ishiguro, T. Ono, M. Imai, and T. Maeda, “Robovie: an Interactive Humanoid Robot,” *IEEE International Conference*, vol. 2, no. 1, pp. 1848–1855, 2002.
(Cited on page 19)
 - [69] T. Salter, K. Dautenhahn, and R. Bockhorst, “Robots moving out of the laboratory - detecting interaction levels and human contact in noisy school environments,” *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication (IEEE Catalog No.04TH8759)*, pp. 563–568, 2004.
(Cited on page 19)

References

- [70] T. Kanda, T. Hirano, D. Eaton, and H. Ishiguro, "Interactive Robots as Social Partners and Peer Tutors for Children: A Field Trial," *Human-computer Interaction*, vol. 19, pp. 61–84, 2004.
(Cited on page [19](#))
- [71] F. Tanaka, J. R. Movellan, B. Fortenberry, and K. Aisaka, "Daily HRI evaluation at a classroom environment: reports from dance interaction experiments," *Proc 1st Annual Conf on HumanRobot Interaction HRI*, pp. 3–9, 2006.
(Cited on page [19](#))
- [72] F. Tanaka, A. Cicourel, and J. R. Movellan, "Socialization between toddlers and robots at an early childhood education center.," in *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, pp. 17954–17958, 2007.
(Cited on page [19](#))
- [73] K. Severinson-Eklundh, A. Green, and H. Hüttenrauch, "CERO: an assistive agent for fetch-and-carry tasks in office environments," *I3 Magazine*, pp. 12–15, 2001.
(Cited on page [19](#))
- [74] N. Mitsunaga, T. Miyashita, H. Ishiguro, K. Kogure, and N. Hagita, "Robovie-IV: A communication robot interacting with people daily in an office," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5066–5072, 2006.
(Cited on page [19](#))
- [75] K. Wada, T. Shibata, T. Asada, and T. Musha, "Robot therapy for prevention of dementia at home-results of preliminary experiment," *Journal of Robotics and Mechatronics*, vol. 19, no. 6, pp. 691–697, 2007.
(Cited on page [19](#))
- [76] C. D. Kidd and C. Breazeal, "A Robotic Weight Loss Coach," in *22nd AAAI Conference on Artificial Intelligence, Proceedings of the*, pp. 1985–1986, 2007.
(Cited on page [19](#))
- [77] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial Intelligence*, vol. 114, no. 1-2, pp. 3–55, 1999.
(Cited on page [19](#))
- [78] T. Willeke, C. Kunz, and I. R. Nourbakhsh, "The History of the Mobot Museum Robot Series: An Evolutionary Study.," *FLAIRS Conference*, pp. 514–518, 2001.
(Cited on page [19](#))
- [79] R. Siegwart, K. O. Arras, S. Bouabdallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, R. Philippsen, R. Piguet, G. Ramel, G. Terrien, and N. Tomatis, "Robox at Expo.02: A large-scale installation of personal robots," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 203–222, 2003.
(Cited on page [19](#))

-
- [80] F. Tanaka, K. Noda, T. Sawada, and M. Fujita, “Associated Emotion and Its Expression in an Entertainment Robot QRIO,” *Entertainment Computing - ICEC 2004*, pp. 499–504, 2004.
(Cited on page 19)
- [81] M. P. Michalowski, S. Sabanovic, and P. Michel, “Roillo: Creating a social robot for playrooms,” in *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pp. 587–592, 2006.
(Cited on page 19)
- [82] M. Jacobsson, “Play, belief and stories about robots: A case study of a pleo blogging community,” in *18th International Symposium on Robot and Human Interactive Communication*, pp. 1–6, 2009.
(Cited on page 19)
- [83] S. Chernova, N. Depalma, E. Morant, and C. Breazeal, “Crowdsourcing human-robot interaction: Application from virtual to physical worlds,” in *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pp. 21–26, 2011.
(Cited on page 19)
- [84] V. Gonzalez-Pacheco, A. Ramey, F. Alonso-Martin, A. Castro-Gonzalez, and M. A. Salichs, “Maggie: A Social Robot as a Gaming Platform,” *International Journal of Social Robotics*, vol. 3, no. 4, pp. 371–381, 2011.
(Cited on pages 19 and 80)
- [85] A. Ramachandran, N. Haven, and B. Scassellati, “Adapting Difficulty Levels in Personalized Robot-Child Tutoring Interactions,” *Machine Learning for Interactive Systems*, pp. 56–59, 2014.
(Cited on page 19)
- [86] D. Feil-Seifer and M. Mataric, “Robot-assisted therapy for children with autism spectrum disorders,” in *Proceedings of the 7th international conference on Interaction design and children - IDC '08*, no. Scassellati 2005, p. 49, 2008.
(Cited on page 20)
- [87] S. Thill, C. a. Pop, T. Belpaeme, T. Ziemke, and B. Vanderborght, “Robot-Assisted Therapy for Autism Spectrum Disorders with (Partially) Autonomous Control: Challenges and Outlook,” *Paladyn, Journal of Behavioral Robotics*, vol. 3, no. 4, pp. 209–217, 2012.
(Cited on page 20)
- [88] H.-L. Cao, P. G. Esteban, A. De Beir, R. Simut, G. Van De Perre, and B. Vanderborght, “A platform-independent robot control architecture for multiple therapeutic scenarios,” in *Proceedings of the 1st international conference on social robots in therapy and education*, pp. 2–6, jul 2015.
(Cited on page 20)
- [89] L. Riek, “Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines,” *Journal of Human-Robot Interaction*, vol. 1, pp. 119–136, aug 2012.
(Cited on page 20)

References

- [90] L. D. Riek, “Robotics Technology in Mental Health Care,” *Artificial Intelligence in Behavioral and Mental Health Care*, pp. 185–203, 2015.
(Cited on page [20](#))
- [91] M. A. Salichs, I. P. Encinar, E. Salichs, Á. Castro-González, and M. Malfaz, “Study of Scenarios and Technical Requirements of a Social Assistive Robot for Alzheimer’s Disease Patients and Their Caregivers,” *International Journal of Social Robotics*, vol. 8, no. 1, pp. 85–102, 2016.
(Cited on page [20](#))
- [92] C. Breazeal and B. Scassellati, “How to build robots that make friends and influence people,” *Intelligent Robots and Systems*, pp. 858–863, 1999.
(Cited on page [20](#))
- [93] R. Barber and M. Salichs, “A new human based architecture for intelligent autonomous robots,” in *The Fourth IFAC Symposium on Intelligent Autonomous Vehicles*, pp. 85–90, Elsevier, 2001.
(Cited on pages [20](#) and [44](#))
- [94] R. Rivas, *Diseño Software de una Arquitectura de Control de Robots Autónomos Inteligentes. Aplicación a un Robot Social*. PhD thesis, Universidad Carlos III de Madrid, 2010.
(Cited on pages [20](#) and [44](#))
- [95] M. Malfaz and M. A. Salichs, “Learning to deal with objects,” in *2009 IEEE 8th International Conference on Development and Learning*, pp. 1–6, Ieee, jun 2009.
(Cited on page [21](#))
- [96] Á. Castro-González, M. Malfaz, and M. Á. Salichs, “An autonomous social robot in fear,” *IEEE Transactions on Autonomous Mental Development*, vol. 5, no. 2, pp. 135–151, 2013.
(Cited on pages [21](#) and [46](#))
- [97] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Press, MIT, 2012.
(Cited on page [30](#))
- [98] B. Widrow and J. C. Aragon, “"Cognitive" Memory,” in *Proceedings of International Joint Conference on Neural Networks*, (Montreal), pp. 3296–3299, 2005.
(Cited on page [35](#))
- [99] R. Wood, P. E. Baxter, and T. Belpaeme, “A review of long-term memory in natural and synthetic systems,” *Adaptive Behavior*, pp. 1–44, 2012.
(Cited on pages [35](#), [36](#), and [37](#))
- [100] P. E. Baxter and T. Belpaeme, “Pervasive Memory: the Future of Long-Term Social HRI Lies in the Past,” *3rd International Symposium on New Frontiers in Human-Robot Interaction at AISB’14*, pp. 1–4, 2014.
(Cited on page [36](#))

-
- [101] J. Phillips and D. Noelle, “A biologically inspired working memory framework for robots,” in *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, pp. 599–604, IEEE, 2005.
(Cited on page 36)
- [102] T. Deutsch, A. Gruber, R. Lang, and R. Velik, “Episodic memory for autonomous agents,” in *2008 Conference on Human System Interactions*, pp. 621–626, Ieee, may 2008.
(Cited on page 37)
- [103] R. Rivas, A. Corrales, R. Barber, and M. A. Salichs, “Robot skill abstraction for ad architecture,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 6, pp. 415–420, IEEE, 2007.
(Cited on page 45)
- [104] M. Malfaz, Á. Castro-González, R. Barber, and M. A. Salichs, “A Biologically Inspired Architecture for an Autonomous and Social Robot,” *IEEE Transactions on Autonomous Mental Development*, vol. 3, no. 3, pp. 232–246, 2011.
(Cited on page 46)
- [105] A. Schatten *et al.*, *Best Practice Software-Engineering*. Springer Spektrum, 2010.
(Cited on page 48)
- [106] K. Beck *et al.*, “Manifesto for Agile Software Development,” 2001. [Online. Last accessed: 2015-02].
(Cited on page 52)
- [107] Google Inc., “Google C++ Testing Framework.” [Online. Last accessed 2015-02].
(Cited on pages 53 and 63)
- [108] A. Kolawa and D. Huizinga, *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press, 2007.
(Cited on page 54)
- [109] M. Quigley, K. Conley, B. Gerkey, J. FAust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, “ROS: an open-source Robot Operating System,” *ICRA*, vol. 3, p. 5, 2009.
(Cited on page 56)
- [110] Willow Garage, “PR2 - Robot for research and innovation.” [Online. Last accessed: 2016-08].
(Cited on page 56)
- [111] Aldebaran Robotics, “Nao robot.” [Online. Last accessed: 2016-08].
(Cited on page 56)
- [112] Google Inc., “Google C++ Mocking Framework.” [Online. Last accessed 2015-02].
(Cited on page 63)
- [113] Python, “Unit testing framework (Python).” [Online; last accessed 2015-02].
(Cited on page 67)

References

- [114] W. Garage, “Rostest framework (ROS).” [Online; last accessed 2015-02].
(Cited on page [67](#))
- [115] M. A. Salichs, R. Barber, A. M. Khamis, M. Malfaz, J. F. Gorostiza, R. Pacheco, R. Rivas, A. Corrales, E. Delgado, and D. García, “Maggie: A robotic platform for human-robot social interaction,” *2006 IEEE Conference on Robotics, Automation and Mechatronics*, pp. 1–7, 2006.
(Cited on page [80](#))
- [116] A. Ramey, J. F. Gorostiza, and M. A. Salichs, “A Social Robot as an Aloud Reader: Putting together Recognition and Synthesis of Voice and Gestures for HRI Experimentation,” in *HRI 2012*, 2012.
(Cited on page [80](#))
- [117] R. Pérula-Martínez, E. Salichs, I. P. Encinar, Á. Castro-González, and M. A. Salichs, “Improving the Expressiveness of a Social Robot through Luminous Devices,” in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts - HRI’15 Extended Abstracts*, pp. 5–6, ACM Press, 2015.
(Cited on page [82](#))
- [118] E. Senft, P. Baxter, J. Kennedy, and T. Belpaeme, “SPARC: Supervised Progressively Autonomous Robot Competencies,” in *Social Robotics: 7th International Conference, ICSR 2015, Paris, France, October 26-30, 2015, Proceedings*, vol. 1, pp. 603–612, Springer International Publishing, 2015.
(Cited on page [83](#))
- [119] E. Senft, P. Baxter, J. Kennedy, S. Lemaignan, and T. Belpaeme, “Supervised Autonomy for Online Learning in Human-Robot Interaction,” *Pattern Recognition Letters*, 2016.
(Cited on page [83](#))
- [120] R. a. Bartle, “Hearts, clubs, diamonds, spades: Players who suit MUDs,” *Journal of MUD research*, vol. 1, no. 1, p. 19, 1996.
(Cited on page [92](#))