Needs-based Companion Dog

José María Blanco Calvo

Delft University of Technology
Faculty of Electrical Engineering,
Mathematics and Computer Science
Man-Machine Interaction Department

Polytechnic University of Madrid
Higher Technical School of
Telecommunication Engineers

Needs-Based Companion Dog

by

José María Blanco Calvo

A thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Knowledge Engineering

presented at

Delft University of Technology,
Faculty of Electrical Engineering,
Mathematics and Computer Science,
Man-Machine Interaction Group

September 2007

Faculty of Electrical Engineering, Mathematics and Computer Science
Section Man-Machine Interaction
Delft University of Technology
Mekelweg 4
2628 CD, Delft

Needs-based Companion Dog

José María Blanco Calvo
Student number: 1294148
E-Mail: josemaria.blancocalvo@gmail.com

Abstract

Needs-based Companion Dog
José María Blanco Calvo (1294148)
Man-Machine Interaction Group
Faculty of EEMCS
Delft University of Technology
September 2007

This thesis describes a project carried out at the Man Machine Interaction research group which is part of Mediamatics at the Delft University of Technology in the Netherlands. *Needs-based companion dog* wants to open new ways of researching in the interaction between robots and humans. In particular the project has focussed on creating a complex personality model for a robotic companion dog.

The societal relevance of intelligent robots is increasing nowadays covering a wide range of applications: from entertainment robots such as conversational partners, soccer players, or companion dogs, to robots that provide independent living support for elderly users in basic activities such as mobility or household maintenance. Security tasks can also be performed by surveillance robots like watchdogs. Robots have shown their utility even in medical domains by aiding the diagnosis and therapy of diseases as autism.

After studying some existing models of artificial personalities, we propose an extension of the *nPME* model, constituted by *needs, Personality, Mood* and *Emotions*. Our main contribution is the use of *needs* (physiological needs, safety needs, and love & belonging needs) as the main forces that control coherent behaviour. We set the satisfaction of the needs always in combination with incoming external events as the basis to evaluate the coherence and realism of the behaviour. Personality and emotions also play an important role in the model so special effort has been put on them to improve the realism of the scene.

Technically we have used a robotic dog called AIBO (made by Sony) as the hardware platform, which was managed by a laptop through a wireless connection. The software for the main control of the system has been implemented using Java. The URBI platform was used to provide the physical control of the robot. The current implementation of the *nPME* model comprises two highly coupled expert systems built using JESS.

Several experiments were carried out to test the validity of our approach. Among them, we did a comparison with the commercial model of Sony, *AIBO's mind software,* and an user study with 24 people to test the realism and coherence of the model for a robotic companion dog. Although results suggest that further work should be done in improving reaction times and some particular actions, they also show that our model was successful in showing a convincing behaviour and coherent emotional responses as a robotic companion dog.

Se admiten en el personaje todas las contradicciones, pero ninguna incoherencia, y en este punto insistimos particularmente porque, al contrario de lo que suelen preceptuar los diccionarios, incoherencia y contradicción no son sinónimos. Es en el interior de su propia coherencia donde una persona o un personaje se van contradiciendo, mientras que la incoherencia, por ser, más que la contradicción, una constante del comportamiento, repele de sí a la contradicción, la elimina, no se entiende viviendo con ella. Desde este punto de vista, aunque arriesgándonos a caer en las telas paralizadoras de la paradoja, no debería ser excluida la hipótesis de que la contradicción sea, al final, y precisamente, uno de los más coherentes contrarios de la incoherencia.

La caverna
José de Sousa Saramago

All the contradictions are admitted for every character, but none incoherences, and at this point we insist particularly because, as the opposite as the dictionaries used to say, incoherence and contradiction are not synonymous. It is inside of its own coherence where a person or a character is going contradicting himself, while the incoherence is, rather than contradiction, more like a constant of behaviour, it repels from itself the contradiction, it removes it, it cannot be understood living with that. From this point of view, although we are running the risk of falling down in the tissues of the paradox, it should not be excluded the hypothesis of that contradiction is, at the end, and precisely, one of the most coherent opposites of the incoherence.

The Cavern
Jose de Sousa Saramago

Acknowledgements

First of all I want to thank my supervisor Leon Rothkrantz for guiding me through the process of creating such a complex project like this. He always put me in the right direction in some crucial moments with very good advice that has given this work the usefulness that it deserves. Very important in this process has also been the continuous help of Zhenke Yang that provided me the practical information that I needed to build a good system in addition with his invaluable advices about the report. Thanks to them, I am proud of the work we have done, and hopefully it will be the seed for upcoming research.

I also want to thank Charles Van der Mast, although he has not been involved directly in the project. His lectures about usability and educational software have opened a new point of view when creating a software product like this. Together with him, I want to thank all the people in the department of Man-Machine Interaction to welcome me as one more Dutch student, showing me that TU Delft is also home. Among them, special thanks to Bou Tsing that also carried out a project with AIBO. He was always willing to help me when some difficulties appeared. Moreover, I want to mention Lorena Cornejo and Daniel González that were loyal useful witnesses when my experiments took place. It is necessary to thank also Fabienne Jesse and Nico Langeveld for their advices during the evaluation stage. Hani Al-Ers was crucial in development of the multimedia content for the videos during several long days, as well as Jose Moar, that was a great camera man.

It is very important to mention my family, unconditional support during these two years of living far away from home. Some moments have been great, but sometimes life becomes harder and there is where my family always encouraged me to keep fighting.

Of course, thanks to my friends from Aranda de Duero and from the hall-residence called San Juan Evangelista in Madrid, because it was always a pleasure to exchange different opinions about how life is constituted, reinforcing the imagination and analysis of what a human being is and therefore his way of thinking. Ideas are the most important value in an engineer and we all have been promoted them for many years. Also their support brought me a piece of home to this great country avoiding losing my way.

Essential is also to thank all the people I have met during my two years in Netherlands. Erasmus students, specially the ones that came from Galicia, Madrid (including Ávila and Ciudad Real) and Cataluña because we have shared great moments during this time and it will be impossible to forget them. Special thanks to Sandra, a great person that Delft brought, because she has provided us so many joys and always an unconditional support in the most stressing moments, showing me the power of design in building ideas. I want to thank also to my flat mates in Delft because they welcomed me as a part of their home and they have let me see how real Dutch student life is from inside.

Table of Contents:

Index of figures

Index of tables

1 Introduction

Artificial Intelligence, as stated by John McCarthy, is "the science and engineering of making intelligent machines, especially intelligent computer programs". [1] However, the idea of intelligence is still incomplete. Although we understand some intelligent processes in the human mind, there is more that remains unknown. Taking this into account, McCarthy defines *intelligence* in [1] as "the computational part of the ability to achieve goals in the world".

Normally, to achieve even a very simple goal (*e.g.: open a door*), many different abilities are implied: recovering information from the world (*where the door is*), reasoning about that (*how a door is opened*), creating an action plan taking into account the environment (*how I can reach the door*), reasoning about the circumstances and even the consequences of performing this plan (*there is anybody in other side of the door*). Later this plan has to be executed and of course, some unforeseen events can occur during that execution. Work has already been done in that direction. For example, one artificial model that fits with this description is the one proposed by Wooldridge [2]: *BDI agent* [1], where an agent is simply an encapsulated computer systems that is able of perform autonomous actions to achieve its design objective when is placed in a certain environment. Using this information a robot can be defined as an artificial agent that acts in the real world.

David Calkins, the president of Robotic Society of America, thinks that robots will be extended as much as computers sooner than we expect. In fact some worldwide known companies such as Philips or Sony have developed some interesting products (i-Cat, Aibo, Qrio, etc…) that promise to be the seed for further research. In this context, robots seem to be an active part of our future, interacting actively with human beings in everyday life. In this way it is absolutely necessary to establish well known models of interaction between artificial beings and people.

This project focuses on the idea of developing natural interaction between man and machine. Since man interacts mainly with other people, the logical step is to develop an artificial brain based on human personality. The artificial brain must act coherently with the environment, drawing conclusions according to the different situations that can arise during a day. That is a very difficult issue but at the same time it lets us judge that interaction more accurately. To achieve this, it is necessary to create a personality model capable of reasoning about a certain situation to perform complex actions in every context. So there must be a whole coherence between events, actions, and emotions.

[1] BDI agent: This agent has *beliefs* (how the world is modelled), *desires* (goals to achieve) and *intentions* (actions to take in order to achieve the goal).

This thesis, 'Needs-Based Companion Dog' contains an effort to create an artificial personality encapsulated into an agent that is able to react appropriately in our world. To test if the model works, a robotic dog called AIBO, developed by Sony, will be used. AIBO can be connected and managed from a normal computer via wireless connection.

This project took the seed from "Personality Model for a Companion AIBO" by Iulia Dobai [3] which attempted to make AIBO show simple reactions to certain stimulus based on a personality model. The goal of this work is to extend this research to make AIBO show complex behaviour, combining sets of actions and emotions, depending on the kind of situation AIBO encounters.

1.1 Motivation

According to the summary prepared by EURON (European Robotics Research Network), about 1.9 million units of robots were sold for domestic use and more than 1 million units for entertainment and leisure up to end 2005. The service robots for personal and domestic use were mainly domestic (household) robots, including vacuum cleaning, lawn-mowing robots. Entertainment and leisure robots include toy robots, hobby systems and education and training robots. The market for handicap assistance is still small, but experts think that this will double in the next four years. They also agree to the increase in importance of the market for robots for personal transportation, home security and surveillance. [4]

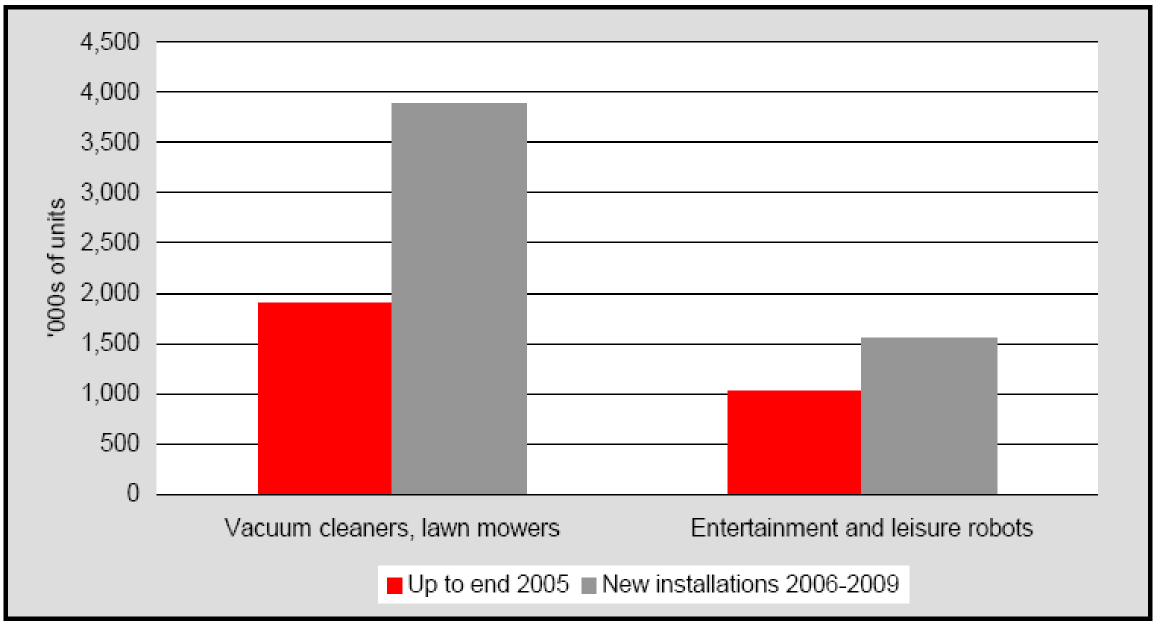


Figure 1: Service robots for personnel/domestic use. Stock at the end of 2005 and projected installations in 2006-2009

The projections for the period 2006-2009 estimate that about 5.6 million units of service robots for personal use will be sold, where domestic robots of all kinds could reach 3.9

million of units and entertainment and leisure robots about 1.6 million units, where most of which are very low cost [4]. Moreover, according to [5], Japanese industrial firms are racing to build humanoid robots to act as domestic helpers for the elderly, and South Korea has set a goal that 100% of households should have domestic robots by 2020.

Looking at this data, there is a high probability that in the future complex robots will be part of our daily routine. One big challenge arises from this point: how to model the interaction between people and artificial beings? The answer seems to be theoretically simple: the interaction between human beings and robots must be simple and natural. Why? The answer could be 'empathy'. This term is defined in *WordReference.com* as: understanding and entering into another's feelings. Empathy is a very basic issue in human relationship. This capability allows us to imagine things as simple as how a person feels when she is sick. With empathy, behaviour becomes more understandable and also more predictable, so initial reticence will likely disappear sooner. So two basic premises, 'ease of use' and 'natural-humanistic behaviour', will help both, humans and robots: first, humans can handle easily routine tasks using robots daily (cooking, cleaning, do shopping, taking care of pets, of children…); and second, if the interaction becomes really natural, the number of 'home' robots will increase exponentially.

One of the pioneers that realized this need of integrating robots into human environments is the company Sony. Sony believed that the best way to make people overcome initial fear of complex robots is to develop entertainment robots. Now, the creation of autonomous robots for entertainment seemed like the most challenging job for researchers but also proved to be a major door opener for future technologies and research platforms [3]. AIBO [6], a robotic dog, is one of the robots that accomplished some initial goals. It was designed by Sony to run around the house and bring a little bit of excitement in a home environment. Since everyone who plays with AIBO shows a wide smile, we can say that Sony has achieved one of its goals.

Figure 2: Old woman playing with companion robot

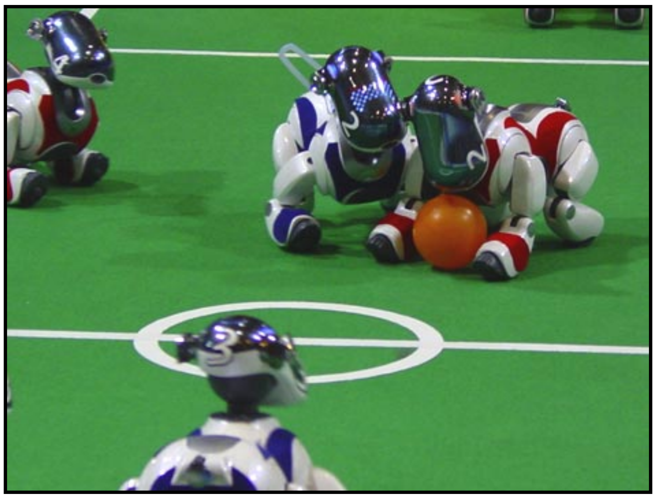


Figure 3: AIBO playing soccer in the RoboCup

The applications of domestic, personal and social robots are very extensive in our society. Some scientist together with psychologists conducted tests on elderly and people with disabilities interacting with AIBO. Studies conducted by Masahiro Fujita [7] [8] asses that AIBO seems to be a good partner with users, having a positive effect on their emotional state. Some very recent experiments studied the acceptance of elderly towards social robots [9] that can support them for an independent living. This may mean supporting basic activities, mobility, household maintenance, monitoring tasks or providing companionship. Other researches show that robots can have a very positive impact in the diagnosis and even in the therapy of autism disease, due to the capability of a robot to repeat some behaviours (to aid gathering data), to record patient responses and also because of their capability of generating high degree of motivation and engagement in children (including those who are unwilling to interact socially with human therapists). Besides these applications, domestic robots can provide other facilities to the users such as taking care of children or pets, surveillance domestic robots (e.g.: watchdogs [10]), conversational partners, reading aloud assistants, rescue robots or even robotic soccer players [11][12].

Figure 4: AIBO used for medical applications with children

Despite the efforts done by some companies, there is much work left. Entertainment complex robots are not as widely used as expected because there are very expensive for the most of families. Companies invest much money in develop their own systems, but even though some of them allow the user to program those robots, the code implemented to simulate the behaviour is normally closed, so no other people can improve their model of conduct. That is one of the reasons why the development of the commercial robots is so slow. Moreover, the models of behaviour normally are quite complex and therefore very difficult to modify or to combine with other techniques. Thus the most of the time a programmer has to start from scratch.

1.2 Project Overview

As stated above there are some handicaps to impulse a fast development of social robots. In this thesis we make an effort to solve some of those weaknesses through the development of a new behaviour model for a robot.

AIBO is able to show emotions and according to the information Sony provides, it can learn and modify its behaviour depending on the life AIBO lives with its owner [6]. This seems to be exciting, but the software created by Sony is not open-source, which means that it cannot be read and of course modified. The consequence is that researchers who want to use AIBO to conduct different experiments are not allowed of modifying Sony's code for their purposes. Then, the utility of AIBO for research despite being potentially very high, becomes quite limited. Consequently we want **to develop a behaviour model that is understandable for engineers, predictable and can be modified easily**. Thanks to this, the personality model will be susceptible of being adapted to different researching purposes in an easy way, because for example the study of elderly reactions or babies reactions with AIBO will need different AIBO's behaviours. Besides, if there is something wrong or strange with AIBO's behaviour, the reason will be found through the study of the personality model or the source code. The use of open-source platforms has another good consequence: the code of this project will be useful for other robotic systems with only a few changes. Not only Sony robots will be able to use our model. So this opens many possibilities for the future.

Moreover we want **to improve the viability of the model developed in "Personality Model for a Companion AIBO"** [3], and check if our approach determines a well-established coherent personality. This is basic to continue with our work that will provide AIBO with a system to choose appropriate actions in each moment, depending on the context. The aim is to make AIBO react coherently in different situations always according to its own personality.

The third goal that this thesis pursues is **to have a flexible system able to adapt to other technologies**. This means that the utility of the initial behaviour model can be increased by adding some new modules. For example, a new *learning module* can make AIBO mature depending on the consequences of the actions it performed in past situations; a new model based in *fuzzy logic techniques* to show emotions can be also added, what will give a more realistic approach since psychology is not a deterministic science. Another example could be the use of genetic algorithms that can be used to add new sets of actions when unrecognized stimulus occurs. Because of this, a big effort on flexibility has been one of the main points that conditioned the design.

These project uses very different disciplines of knowledge: *man machine interaction, software engineering and hardware managing,* and *psychology*, so it has to face several interdisciplinary difficulties. First one resides in the fact that there are not much previous work about long term interaction between a person and a companion robot, and the principles we use in the model have not been experienced enough times in the past. Moreover, models of human brain and human personalities are not complete at all. There are some accepted principles, but these do not define a whole person. About the hardware itself, Sony finished its production of AIBO units. Future applications seem to be lost. However, our system is going to be developed in URBI [13], an open source language that can be used with many other robots. URBI is a universal platform. Thanks to this, if even AIBO robot is not available, our system will be able to be adapted in different robotic platforms.

To give a whole overview of the project, at the beginning we studied carefully the thesis "Personality Model for a Companion Dog" [3]. Initially this was our base, so we tried to reuse the main ideas and even pieces of code when it was possible. In "Needs-Based

Companion Dog" we modify the previous personality model to make it more suitable for implementation, taking the ***needs*** of a life being as the main forces that drive behaviour. Therefore this new model is the most important part of our thesis.

Once we have translated the personality model into computer code and the robotic 'brain' shows a coherent behaviour depending on the context and its personality, we will start to create sets of actions that AIBO will perform in real world according to that behaviour.

Later we will focus in the flexibility of the system, developing a new module that will constitute the main control of the overall system. This is the one that, for example, will join the emotions created by the brain and the hardware to express them in the right moment. That is one of the main goals of the implementation together with the personality model and it will be also the basis to add new modules with other capabilities in the future.

In the last stage, we will combined all the elements allowing AIBO reasoning about different stimulus and according to its personality, it will choose an adequate set of actions and emotions to perform in the real world. The goal of this stage is to make the visible behaviour of AIBO be totally coherent with the context where it is acting and its own internal state.

When all these steps are done, the evaluation part will take part following the right side of the V method. All the modules of the software are tested alone as integrated parts of the whole system. When these verifications are successful the system is connected to AIBO robot to test the system as a whole. If the integration goes right, the last stage works entirely with AIBO robot, adjusting some variables to get a continuous and natural behaviour, to validate the whole system.

The final part is the evaluation of our approach. Several experiments will test the validity of the system from different points of view. First a technical evaluation will give an analysis about the advantages and shortcomings of the architecture of the system together with the tools used for the implementation. Then a comparison between the commercial system developed by Sony (in AIBO Mind's software) will present the similarities and differences with our model, showing some drawbacks of the former that our system will overcome. Finally four detailed scenarios will be proposed and a video of AIBO performing on them will also be recorded. Using that video another experiment will be carry out as a manner of survey with more than twenty people. The aim of this survey is to check the reactions and acceptance of those people when watching our companion AIBO in different situations that can arise in a daily life of a real companion dog. Their answers will be used to give some light about the success of our approach in terms of subjective impressions about what they think is a proper behaviour for a robotic dog. Some statistics will be given to show their tendencies that can aid for a future development.

Using all the information during the evaluation stage, some conclusions about the whole work will be given. These will help us to propose several recommendations for the improving of the current system together with new ideas for the creation of more accurate artificial models of mind.

1.3 Problem Definition

To set the focus of our thesis we gathered data from previous researches about artificial mental models for social robots and ways of interaction between people and robots in domestic environments [14,15]. This preliminary user study included also information about domestic robot applications to increase the usefulness of the project. Besides, ethologic information about dogs was used [16] to get ideas of the behaviour that a real domestic dog usually presents. Thanks to all that information we delimited the scope of this thesis, setting the first group of requirements.

Our robotic dog, AIBO, must behave as an autonomous companion dog what means that it will act in a domestic environment facing some common situations (sleeping, eating, playing, etc) without supervision. Its behaviour must be similar to a real dog, so realistic sounds and realistic movements are required. Since a real dog reacts to certain stimulus (food, owner, friends, etc) our robotic dog must also react to events that wrap those stimuli (battery levels, owner image, owner sounds, etc). The main forces that drive a real dog are the needs (need of food, need of feel safe, etc), so our robotic dog will be also driven by a model of needs (need of battery, need of safety, etc). Its behaviour when satisfying those needs should be similar to the shown by the real dog. Again, as a real dog show emotions in certain situations (her owner leaves home, her owner appears at home, etc.) our robotic dog must also express those emotions in appropriate contexts.

To get such a complex behaviour, an artificial personality model based on the nPME model will be designed and implemented in a prototype to test the validity of our approach. Besides, to implement the model we need a whole system able to translate the external stimulus to computer events able to be processed and used in the personality model. This system also has to provide a graphical user interface that let the user to monitor all the personality variables in the system. In addition the design of the system must provide enough flexibility to allow future users change one or another component. In this way we assure the continuity of the system for future research and let other researches to include their own mental models if necessary.

Once the system is built, some experiments will be carried out to test the correctness of our approach. After a technical evaluation, an experiment with people must be done to get ideas about the impressions that such a robotic dog may cause. This extensive user study, as a manner of survey, has to be done with a working prototype using the robotic dog AIBO. The results of the survey will be used to highlight the advantages and the shortcomings of the mental model to draw some conclusions about our work. This data can also provide some good ideas for future recommendations.

1.4 Report overview

This report has 7 chapters plus appendix and references. The first chapter (the introduction) gives an overview of the project and explains the reasons that motivated it. Goals pursued in the thesis are pointed out, and the problem definition is presented. The rest of the thesis is structured as follows:

In chapter two, we will study some necessary background to frame the problem and to infer some possible solutions. We will talk about the robot AIBO and focus on two existing personality models that were developed for this machine. In addition, three new ideas of models for the project are proposed. Some notes will also be included about important aspects of programming robots that can be useful to understand the final system.

The third chapter contains a more in-depth analysis of the previous psychological models in relation with more technical issues to extract the list of requirements that the system has to fulfil. These requirements form the basis of the design and the implementation of the final system.

The fourth chapter presents the complete set of functionalities that the system must provide together with UML use case diagrams. It also gives the necessary information to understand the world that the robot can perceive. Taking this into account a complex behaviour is depicted to end up with the architecture of the final system.

The implementation details of the system are presented in chapter five with an introduction to the tools used for it: URBI, Java, Jess. All the different modules of the system are explained together with some UML diagrams that clarify the contents.

Chapter six is dedicated to the evaluation of the system. After a technical discussion two experiments are proposed to test AIBO behaviour. The results of the evaluation are also included in this chapter, to check if our approach has been correct.

Chapter seven gives the final conclusions along with future recommendations.

Finally, some useful appendixes were added, included to clarify some of the contents of the thesis and to provide additional information that can be useful for future developers.

2 AIBO background

This chapter provides a basic background about AIBO to understand its possibilities and its limitations. The first part explains what AIBO is and its basic features. In the second section there is a more detailed description of AIBO capabilities and different uses. The third section gives an overview of two models already implemented for AIBO's behaviour. The fourth one provides three proposals of different models of personality. One of these models has been chosen to implement in this thesis.

Chapter Overview:

Figure 5: AIBO ERS-7

2.1 What does AIBO mean?

This section gives a brief explanation of what is AIBO and which are its main sensors and actuators.

AIBO is a robotic dog made by Sony. It was developed with the idea of a fully entertainment machine. AIBO is a companion dog and a useful system that can play music, attend to some owner's commands, connect to Internet to send e-mails, etc. The design was made very carefully trying to promote friendliness and amusement from the users. The robot integrates Hardware and Software becoming AIBO in a complex robot that matures according to owner's life.

AIBO receives information from the outside world through his sensors: video camera (a 350,000 CMOS image sensor), stereo microphones in its ears, two distance sensor, and various touch sensors on head, back, chin, paws, acceleration sensor, and a vibration sensor. Those sensors are shown in Figure 6 and Figure 7.

In order to show reactions AIBO is equipped with speaker on its chest, LED Lights (on face, ears and back) and a series of movable parts: head (3 DOF[2]), mouth (1 DOF), legs (4*3 DOF), ears (2*1 DOF) and tail (2 DOF). Apart from this AIBO is also equipped with a wireless card and a blue LED to show its status. For more information regarding the physical capabilities of AIBO please refer to Appendix J.

The centerpiece of AIBO's artificial software is the AIBO Mind 2 software, located on a removable Memory Stick™. This controls AIBO's behavior and the applications that can be used to interact with AIBO via PC or a mobile device.

2.2 AIBO features

This section provides more in detail some of the most important features of AIBO robot according with the information provided by SONY.

AIBO is a companion pet for day-to-day life that will try to entertain and even communicate. It is possible to distinguish in AIBO two types of features: *autonomous*, when it acts as a common dog; and *manual* features that can be used by the owner to make AIBO do other tasks than a companion dog.

[2] DOF = Degrees of Freedom

2.2.1 Autonomous Feature

AIBO's personality is only software and it is located on the Memory Stick. Both personality and memory can be changed and updated. In this way, The AIBO Mind 2 Software can be chosen to present an AIBO from babyhood to adulthood, or a mature AIBO fully educated. Over time, new skills can be added to AIBO by upgrading the AIBO Mind2Memory Stick. However, Sony stopped AIBO production, both hardware and software. Some particular users offer its own 'mind designs' for free via Internet.

Quite autonomous, AIBO can see, hear, feel for itself and walk. Moreover, AIBO has been developed to be able to recognize owner's voice and face. AIBO has a LED-guided face that can reflect a wide range of emotions. This will depend on some external stimuli and a random component.

When AIBO detects that its battery life is low, it will try to replenish its batteries. It will try to find the charging station using pattern recognition to locate a fixed pattern in the station. Once the station has been found it tries to approach and sit down in the correct way to charge its batteries.

It has some accessories like a pink bone or a pink ball. If one is thrown, AIBO will look for it. If AIBO recognizes the object, it will fetch it for you. AIBO's skill level can improve through encouragement from its owner. [17, 18]

2.2.2 Manual Features

There is also a Navigator interface which can be used to instruct AIBO via wireless connection from a normal PC to obtain better heights of performance. With Navigator remote control, obstacle avoidance and recovery time is enhanced.

AIBO enjoys music and being a natural performer; it will play music on demand, and even dance with the right command. AIBO plays internet radio, MP3 or CD. One can make AIBO play a particular CD simply by showing the cover. Moreover AIBO is able to connect wirelessly with other electronic devices, transmitting photos, sound files and messages. AIBO can even record movie clips.

In House Sitting mode, AIBO takes pictures or records sounds from its energy station in response to moving objects, faces, or sounds. When it detects a movement, face or sound, AIBO can be set to snap a picture and notify to the owner by e-mail.

AIBO can adapt to owner's daily routine. Thanks to the Scheduler feature the user can import calendars and have AIBO read the schedules out. There's no need for a clock radio either as Scheduler ensures that AIBO will play music at a programmed wake-up time. [17, 18]

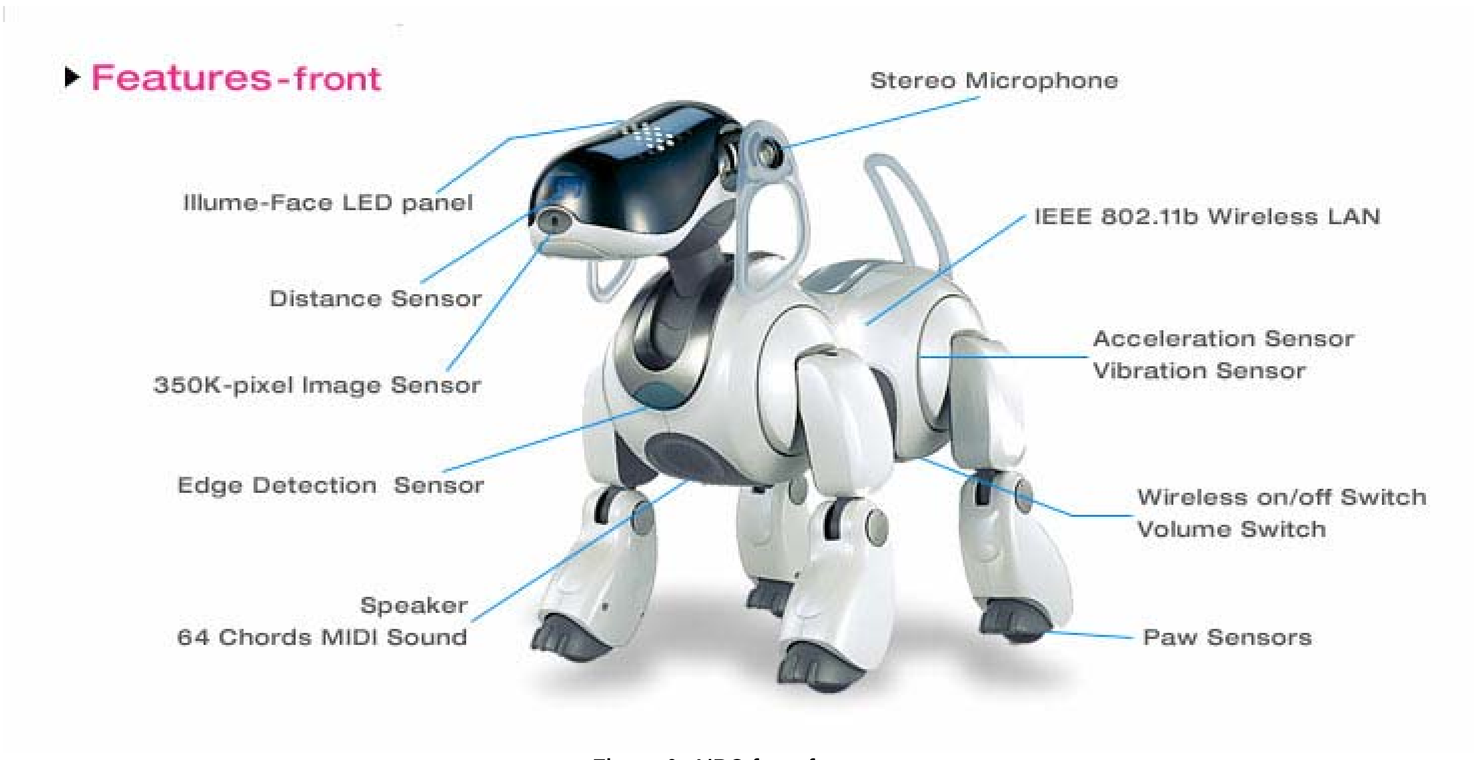


Figure 6: AIBO front features

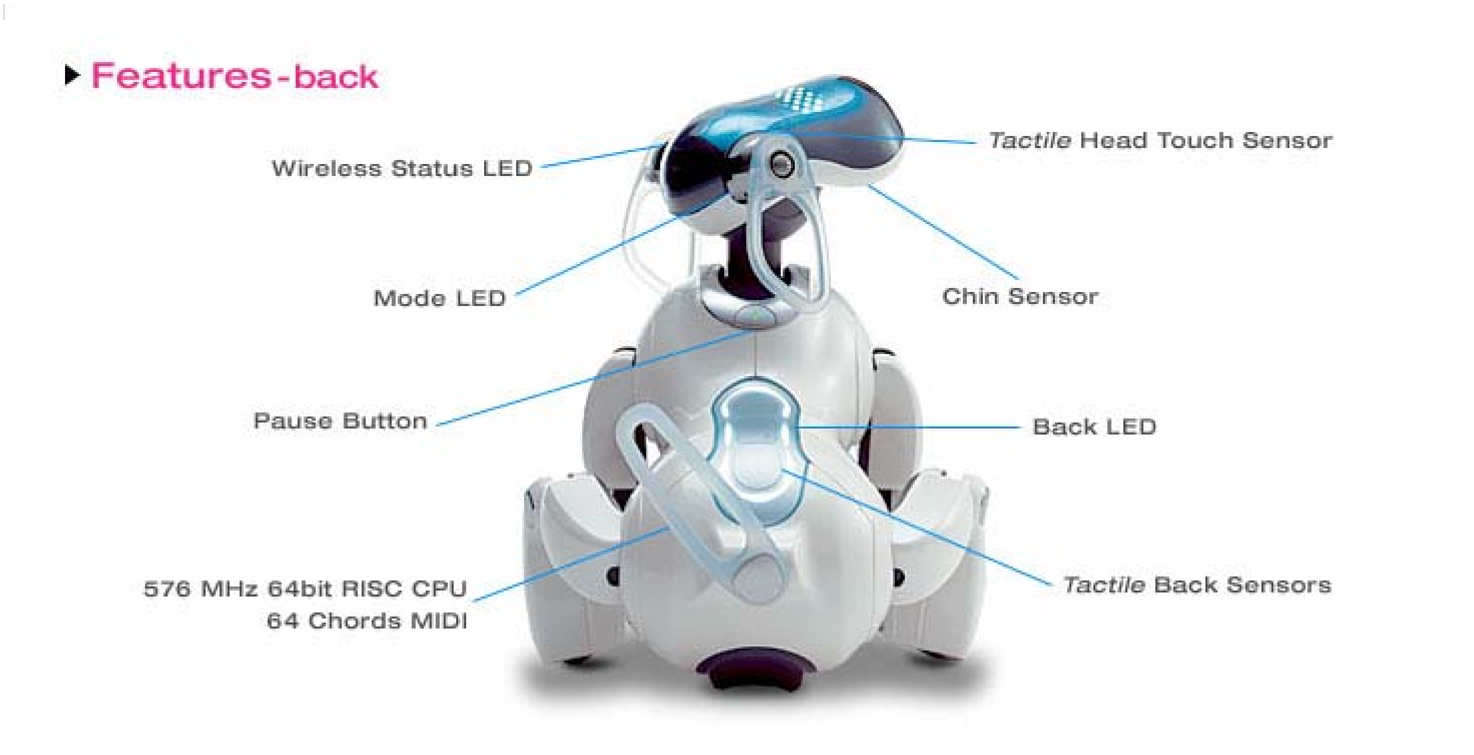


Figure 7: AIBO back features

2.3 Personality Models for AIBO robot

We present here several models for AIBO personality. The first two have been already implemented. The last three models are only our ideas for the final design of our project, based partially in the analysis done of the other two models.

Firstly, it is necessary to say that there is a fundamental limitation for the development of a complete cognitive model: *nowadays researchers still do not know exactly how the brain works*. That is the reason that there are many different models of personality. Issues that have been discussed from the beginning of the psychology (by Freud, Erikson, Fromm, etc…) and they are still under discussion. Since personality is one of the less receptive fields in psychology, because there are some aspects that are not conscious like instincts and unconscious motivations, many aspects in the research of this topic are still in a "pre-scientific" or even in a philosophical stage. Maybe some of the deepest aspects of personality remain unknown forever.

Acknowledging these limitations, the best we can do is to build a model which considers the most important aspects of the personality. 'Important aspects' refers to the main features that all the minds share and can be used to classification purposes. However these features are only a few so they are not able to characterize a whole real mind by themselves.

According to Damasio, "***Emotions are the brain's interpretation of reactions to changes in the world***" [19], so the developed system must characterize a brain that interprets these changes and reacts to them.

In this section we present two attempts to create a solid personality for AIBO. First, it is shown the original model developed by Sony for their commercial minds. The second one was carried out by the Technical University of Delft.

2.3.1 Sony's Behavioural Architecture for AIBO

This section presents the model developed by SONY for its entertainment robot AIBO. This model has been built using ethological models from animals observation and emotional models as motivational drives extracted from human behaviour. [20] The behaviour election is done according to external stimuli and the regulation of its ongoing internal variables. This model has been used in commercial AIBO showing that theoretically the model is rich enough but in the

practical AIBO shows many random reactions and it does not present a continuous emotional state.

Sony has put much effort in developing ethological[3] and emotional models as the basis for a architecture in entertainment robots, where human psychology plays an essential role to improve man-machine interaction.

In [20], Sony stated that there are basically two goals to achieve a natural relationship between a human and a robot:

1. The creation of high-fidelity ethological models to make AIBO behaviour more predicable by people.
2. The creation of motivational behaviour (e.g. emotions) that supports human conceptions of living creatures so it makes easier to bonding between human and robot.

They think that through observational behaviour it is possible to extract suitable descriptions of animal activity that can be mapped onto robotic systems. Besides, human emotions and their similar occurrence as motivational behaviour in animals can help to improve interactivity between man and robot.

I. *Ethological Basis*

From an ethological point of view, they have categorized different animal activities into what they call *Behavioural Subsystems* (e.g. 'investigative', 'epimeletic', '**agonistic**', 'sexual', etc). - The one in bold is the one that is divided in next paragraph.
Each subsystem is also divided into different *Modes* (e.g. '**Defense-Scape Mode**', 'Subordinate attitude mode', etc). And each mode successively into *Modules* (e.g. 'Running-away', 'Crouching', 'Yelping', etc)

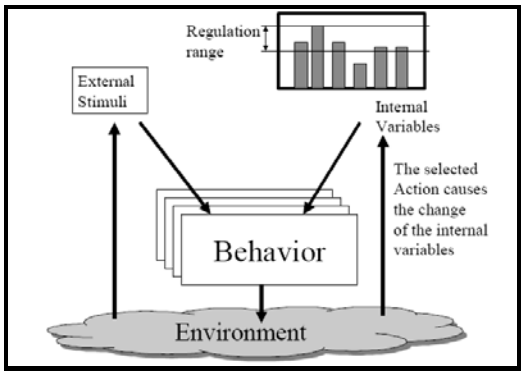


Figure 8: Role of Drives in Behaviour Selection

[3] Ethological: related to animals behaviour in their natural setting.

II. *Emotional Basis*

A particular behaviour will be selected in a given situation. This is done evaluating external stimuli and ongoing internal variables. Both changes in the environment and behavioural actions will make changes in these internal variables, and they should be always within proper ranges ("homeostasis regulation rule" [20]). The action selection is depicted in Figure 8.

Aibo shows 6 basic emotions according to Eckmann's model [21]: happiness, anger, sadness, fear, surprise and disgust. There is one square where it is mentioned the *[P, A, C] space*. That is a 3-dimensional space based in Takanishi's model [22]. It is composed by *pleasant*, that will have a high value if the robot variables are within the regulated range; *arousal*, variable that is controlled by both circadian rhythm and unexpected stimuli; and *confidence* that measures the certainty of recognized external stimuli.

Figure 9 shows how the level of the internal variables motivates both output emotions and external behaviour in AIBO.

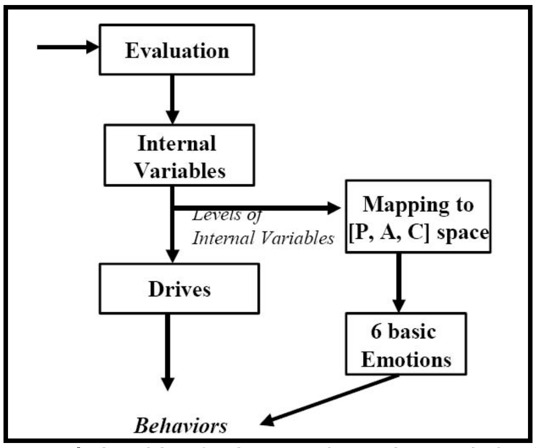


Figure 9: Relationship of Drivers and Emotions to behaviours

III. *AIBO Architectural Implementation*

The **Architecture** of the system combines three main parts:

- **Releasing Mechanism**: computes perceptual results of the environment (visual and auditory mainly) to get its output RM[I] (where I is the detected object).

- **Motivation Creator**: computes next two models to give the output Mot[I]:
  - *Emotional Model:* it includes 6 internal variables: nourishment, moisture, bladder, distension, tiredness, curiosity and affection.
  - *Instinct Model:* it includes also 6 variables that act to keep the 6 internal variables within some bounded values.

- **Action Selection module**: uses Releasing and Motivation outputs to compute a behaviour variable V[I]. This will lead to choose one behaviour. To avoid doubts between elections, the system uses lateral inhibition[4] to get only one (in Figure 12 we can see a multiple layer diagram representing different behaviours). When the computations have been performed to select a proper action command, it is sent to a Finite State Machine where the specific sequences on how to achieve the behaviour are described.

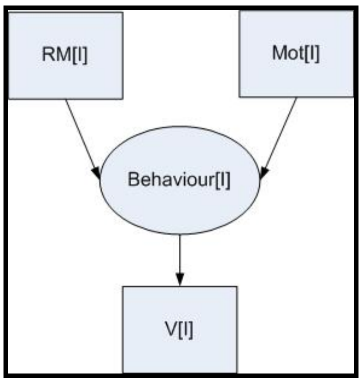


Figure 10: System Architecture

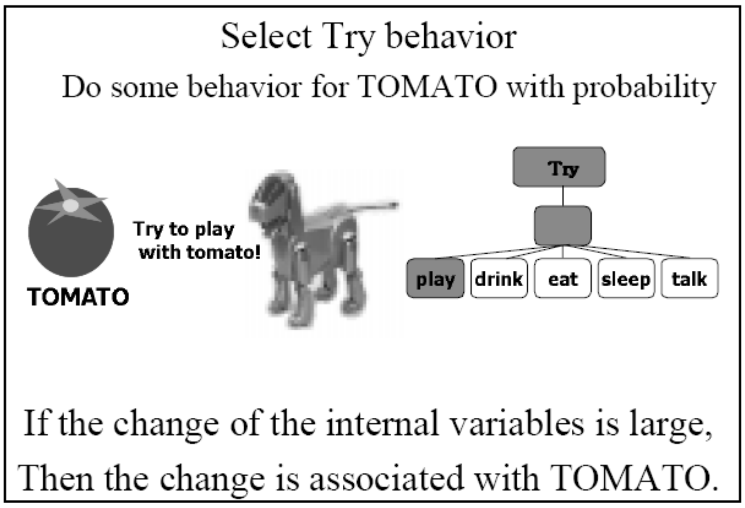

Select Try behavior
Do some behavior for TOMATO with probability
TOMATO
Try to play with tomato!
Try
play
drink
eat
sleep
talk
If the change of the internal variables is large,
Then the change is associated with TOMATO.

Figure 11: Example of behaviour for Experimentation with new Environmental objects

IV. *Emotionally Grounded architecture*

In Figure 12, the extended Emotionally Grounded architecture is depicted. This is used only with symbol acquired behaviour. The system remains the same as in the explanation above, but now it incorporates a new subsystem for unknown objects. In other cases, if the robot finds an unknown object, it has no way to know what it is. However, they have developed a system that makes AIBO modify some of its internal variables when it sees an object that is unknown, so somehow it knows the new object, and it can react to it. The goal is to 'learn' the meanings of the acquired symbols in terms of robot needs. Figure 11 shows conceptually how this system works.

[4] Lateral Inhibition is a process based in biologic neurons to locate accurately the stimulus that makes them react. Despite the fact that every neuron in some area react to the stimulus, the neurons where that was stronger inhibit the rest, making the stimulus be well located.

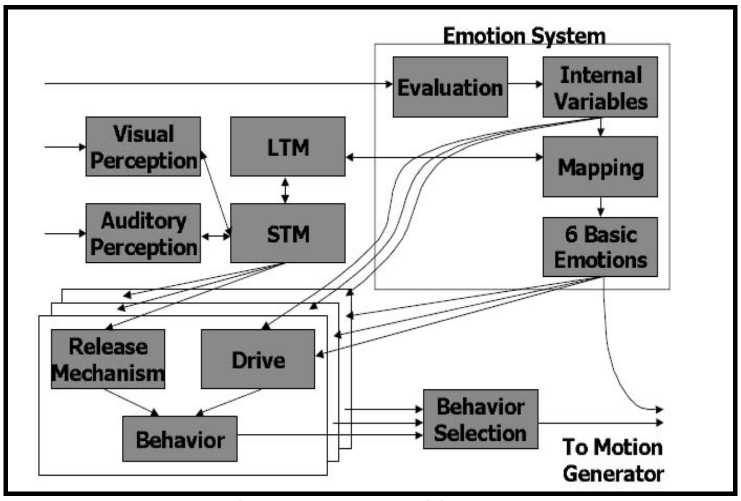


Figure 12: EGO Architecture

2.3.2 nPME model

This section presents the personality model developed in TU Delft by Iulia Dobai in 2005 [3]. This model is based mainly in needs, Personality, Mood and Emotions parameters, so it is called nPME model. This model has been partially implemented in reality showing one action and/or emotion for each estimuli, but not a continuous behaviour. Due to the number of parameters it uses, the implementation presents a problem of combinational explosion.

Since the code of AIBO's behaviour was not open source code (*OPEN-R SDK* [23] by Sony), and the research made in "Emotional AIBO" [34], Man Machine Interaction Group of TU Delft decided to carry out the project "Personality Model for a Companion AIBO". The aim was *to design and implement a mental model for AIBO that represents the basis for showing emotions and actions during interaction with humans in some specified contexts. The mental model developed needs to be presented in a software framework that is flexible and open.* [3]

The 'brain' itself is running in a normal computer and AIBO can be controlled via wireless connection. They used *Java*[5], as the core of the system; *Jess*[5], an expert system[5] shell to do the reasoning; and *URBI*[5] to manage the connection with AIBO and move its joints.

Thanks to this AIBO can get some events, both external and internal, do some reasoning process, show emotional behaviour and perform some basic actions. It is a simple reflex system[5]: from each event, AIBO shows one action and/or emotion.

[5] This term is explained in the glossary section.

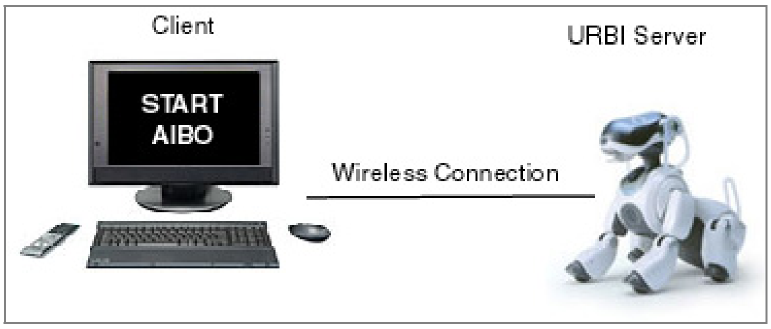


Figure 13: System Architecture

To simulate external events, a graphical interface was implemented to give AIBO direct commands like 'Sit down' or characterization commands like 'Good dog'.

AIBO reasoned combining those events with the *nPME* model to show an appropriate response for every case. nPME model combines Personality, Mood, Emotions an also 'needs' to establish a complex behaviour model. When the response has been computed, some interpreters convert it into URBI commands that are understandable for AIBO's operating system called *Apertos*.

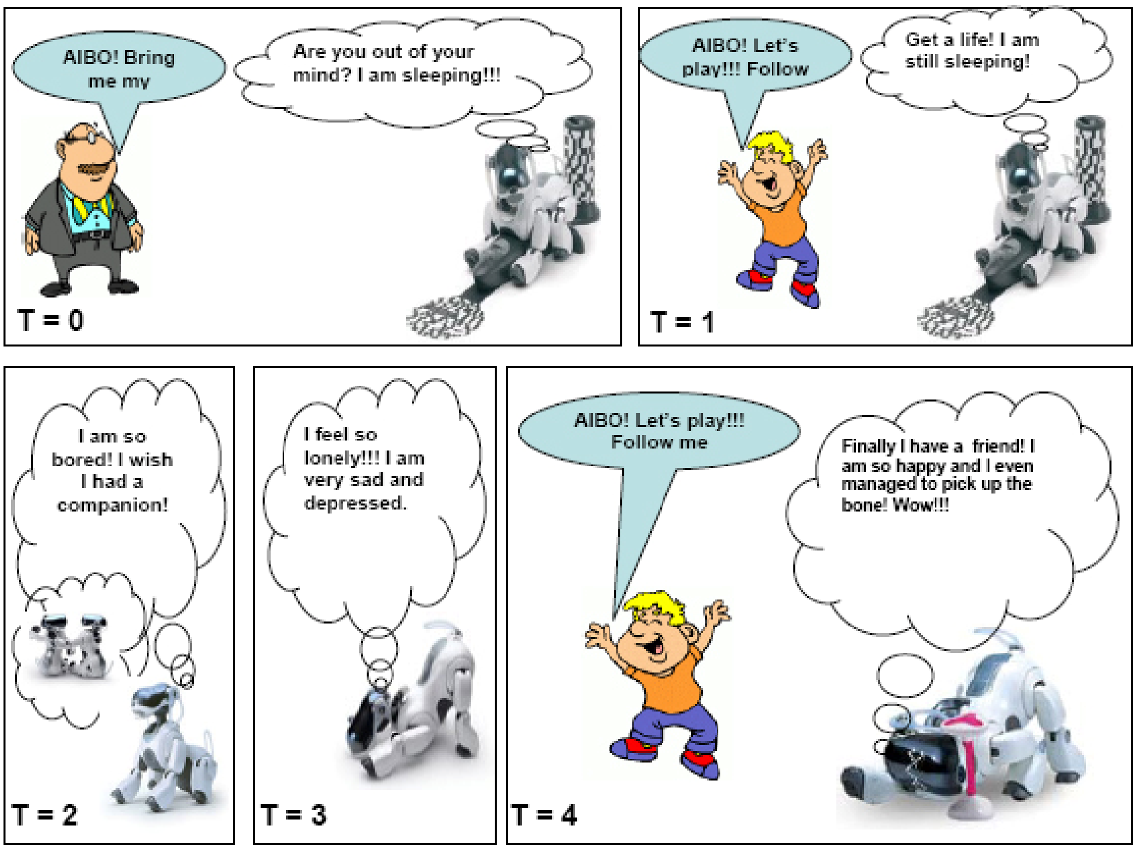

AIBO! Bring me my
Are you out of your mind? I am sleeping!!!
T = 0
AIBO! Let's play!!! Follow
Get a life! I am still sleeping!
T = 1
I am so bored! I wish I had a companion!
T = 2
I feel so lonely!!! I am very sad and depressed.
T = 3
AIBO! Let's play!!! Follow me
Finally I have a friend! I am so happy and I even managed to pick up the bone! Wow!!!
T = 4

Figure 14: Desired behaviour of the companion robot when using the nPME model

I. Personality Model

First of all it is very interesting to give some theoretical definitions of the concepts involved in the construction of this model of personality. These will help the reader to understand the logic of the system.

i) Personality

Zimbardo defines personality as the psychological qualities that bring continuity to an individual's behaviour in different situations and at different times. It is the thread of continuity in an individual in different situations .[3]

Personality can be characterized by 'Traits'[6] that are the stable personality characteristics that are presumed to exist within the individual and guide his or her thoughts and actions under various conditions. To choose these traits the "Big Five" model [24] has been used. It gives five dimensions to personality, according to the paper "Five Factor Constellations and Popular Personality Types", by Leland R. Beaumont. Acronym OCEAN can be used to summarize the "big five".

Table 1: Personality Traits in the Big Five Model (adapted from John – 1999 - and Zimbardo – 2002 -)

Trait	On one hand	On the other hand
Openness to experiences	Originality, Open-mindedness, Inquiring intellect, imagination, curiosity, independence, cultured	Closed to experience, close-mindedness
Conscientiousness (it concerns the way we control, regulate, and direct our impulses)	control,constraint,dependability,cautiousness,perseverance,superego, strength, prudence	Lack of direction, impulsiveness, carelessness, irresponsibility
Extraversion (engagement with external world)	energy, enthusiasm, social adaptability, assertiveness, sociability, boldness, self-confidence	Introversion
Agreeableness (individual differences in concern with cooperation and social harmony)	altruism, affection, conformity, likeability, friendly, compliance, warmth	Antagonism, coldness, negativity
Neuroticism (tendency to experience negative feelings)	negative affectivity, nervousness, anxiety, emotionality	Emotional stability, emotional control

[6] Trait is defined as a distinguishing feature of your personal nature. (Source: www.wordreference.com)

ii) Emotional States

An emotional state is a particular state of mind that is reflected visually by an emotional expression. [3]

Emotional categories proposed by the OCC Model [25] have been used, which categorizes various emotional states based on positive or negative reactions to events, actions and objects.

Table 2: Basic Emotional States - OCC model

Positive	Negative
Joy	Distress
Pride	Hate
Love	Fear
Hope	Remorse
Relief	Anger
Gratitude	Disappointment

OCC model has twelve different emotional states, but in for computational complexity and also because the limitations of the robot, in [3] those states were reduced to 6 basing on Eckmann's model [21]:

Table 3: Emotional Expresion / Emotional States

Expression	State
Joy	joy, pride, love, hope, relief, gratitude
Sadness	distress, remorse, disappointment
Anger	anger, hate
Fear	Fear
Surprise	
Disgust	

In addition to these AIBO will show **surprise** and **disgust.** They are not present in the OCC model mainly because they do not involve much cognitive processing and they do not correspond to reactions. [25]

iii) Mood

Mood is a conscious and prolonged state of mind that directly controls the emotions. Mood is constant for longer time spans than emotions.

The used mood model is based in the proposed by Lang [26] in 1995. Basically it is a two dimensional system with *valence* and *arousal* as axes. Valence refers to good or bad mood, and arousal means intensity of it.

iv) Needs

To characterize an alive being, it is absolutely necessary to talk about its needs. The main force in all the creatures is the survival feeling. It is an unconscious knowledge that makes us fight to get all the necessary things to sustain our inner physiological processes. Every creature needs basically eating, resting and procreating. The complex is the being, the complex are its needs. Humans need also a place to take shelter, clothes to protect themselves against cold, social acceptation to live in community, etc.

These needs make a creature look for satisfy them what usually cause one behaviour or another (*e.g. If a baby is hungry, he will start crying. If a dog is very tired, it can ignore its owner and goes to its bed*). Therefore needs are an essential part for an emotional behaviour model.

Iulia [3] used a need model based on the Maslow's theory, known as "Maslow Pyramid of Needs" [27]. It is the most widely accepted model for needs characterization and also it is simple enough to be implemented. Figure 15 shows that Pyramid.

The idea behind the graphic is that the need which is at the bottom of the pyramid is the first that must be satisfied. On the other side, the one which is on the top will only be satisfied if all the rest under it have been achieved.

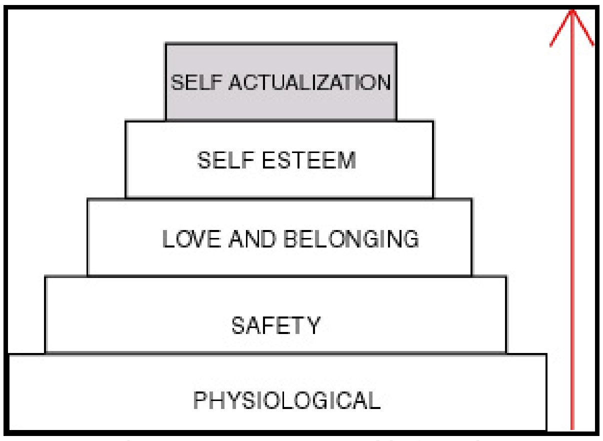


Figure 15: Maslow Pyramid of Needs

The first four categories are called *deficit needs*, and the last one ("self actualization") is known as *being needs.*

These needs were thought for human beings, so in the model for AIBO they have been translated to real needs for a robot of those features. *E.g.: physiological needs will mean need of recharging batteries in AIBO's life.*

Needs have priorities and act as thermometers. Once critical values have been reached by different categories of needs depending on their priorities they lead to different structures of goals, preferences and standards. [5]

v) Goal, Preferences and Standards

It is the moment where Goals, Standards and Preferences are introduced. They provide us a tool to relate the ideas of mind to actions in the real world. Therefore these components will form the bridge between concepts of Personality, Needs and Mood and the material performance of Actions and Emotions.

Summarizing, the definition of the concepts is:

- ***Goal***: Tasks orientated to objectives that are SMART (simple, measurable, acceptable, realistic, time frame). *E.g.: AIBO finds ball.*
- **Preference**: appealing ness to aspects of objects. E.g. AIBO likes more bone than ball.
- ***Standard:*** approval/disapproval regarding actions of agents[7]. Standards can focus on self agent or other agents. *E.g. AIBO likes its owner touches its back, but not if it is an unknown person.*

II. nPME model as a combination of its components

Until now in the design of virtual humans, agents or game characters, two categories of personality models were mostly used: PME models that take into consideration personality (P), mood (M) and emotions (E) and PE models that are based solely on personality (P) and emotions (E). [3]

The design of the nPME model starts with a layered PME architecture. Mood is seen as an intermediate layer between personality and emotions and therefore is influenced by both. Since we want AIBO to act independently in a changing environment we have to take into consideration his individual needs, therefore we included in the layered architecture the needs parameter (**n**PME). By adding needs we also had to make some major changes in the existing architecture and therefore mood is directly influenced by neither emotions nor personality, but it is indirectly influenced by them as a consequence of the evaluation of the goals, preferences and standards in regards with the events that take place. [3]

In a more schematic way, the concepts that will be used in the nPME model and the relationships between them are:

- **Personality**: it does not change through time, so the values for the 5 dimensions of *personality* remain the same through the application.

- **Mood:** it will be the result of the constant traits of *personality* and changes in *emotions*. *Mood* is constantly changing in small amounts.

[7] ***Agent***: is every system, biological or mechanical, that can perceive some stimuli from the environment with 'sensors' (eyes, camera, microphones, etc) and also can act in this environment with its effectors (hands, paws, mouth, mechanical tail, etc).

- **Emotional expressions** and **states** become the one single unit in the system. These are: *happy, sad, angry, fear, surprise, disgust.*

- **Needs** are extremely important. These together with *personality* will define the initial *mood*, and will create some *goals, preferences, standards*. Insofar the needs are satisfied, these will change and new GPS will be produced. So indirectly needs will also be conditioned by *personality*.

In Figure 16 it can be seen that every time there is a new set of *needs* present in the system, a small engine will be activated to update values for *goals, preferences* and *standards*. If an event occurs (by event we understand a registered occurrence that is detected by AIBO) the response engine is triggered and based on the evaluation of *goals, preferences* and *standards* and on the current *mood* a new *emotional state* and a set of actions results. [3]

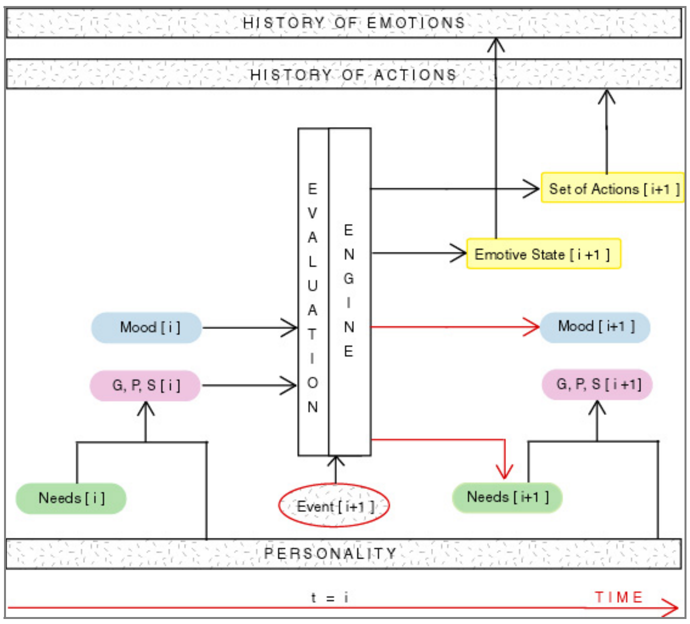


Figure 16: nPME model at moment t=i [1]

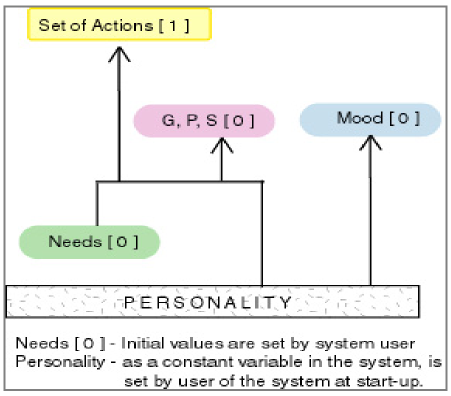
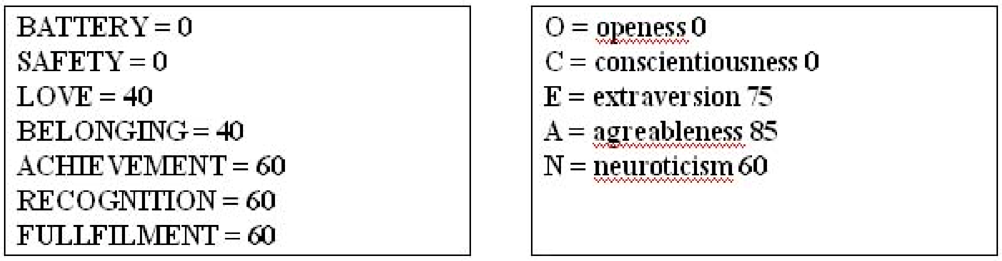

Set of Actions [1]
G, P, S [0]
Mood [0]
Needs [0]
PERSONALITY
Needs [0] - Initial values are set by system user
Personality - as a constant variable in the system, is set by user of the system at start-up.
Figure 17: nPME Model at moment t=0 [1]

Figure 17 presents the initialization of the entire system at moment t=0. At this moment we have to take into consideration the possibility that no event happens in a certain amount of time, therefore based on the personality of AIBO and its needs we decide on a few actions randomly chosen from the list of possible actions. It is important to mention that even when there is an absence of activity during a specified amount of time, a special event will be generated. (*e.g. the absence of any human for prolonged amounts of time in the vicinity of an AIBO represents an event that will lead to an increase of the need for love and belonging and a decrease of the positive value of mood with a precise amount.*) [3]

Following is an example to clarify how nPME model works taking into account Figure 18:

> *When AIBO is initialized, the values for Personality and Needs are given in two text files. One trait of personality is 'Extraversion' and the initial value will be 80 out of 100. In needs txt file there is one called 'Love'. So in the first step, from both data, Mood will be set in 60 out of 100 (what means good mood) and because of the Extraversion high value, Love value will also be increased. A this point some Preferences and Standards will be created. Taking into account again the high level of Extraversion, the system can set two Preferences saying that AIBO will like to meet AIBO-Friends and ALL-Humans. Also because of Extraversion Value, AIBO will like the action of Touching AIBO's Head by ALL-HUMANS (that is an Standard).*
>
> *In next step with this data, if AIBO's head is touched by a Human (these is called an 'Event'), thanks to the Standard, AIBO will react and show Happiness emotion and one friendly action like wagging the tail.*
>
> *On the contrary, if Extraversion value had been low and also its need of love, both Preferences and Standards for ALL-Humans would be lower what means that AIBO had reacted aggressively if its head had been touched by one human.*



```
BATTERY = 0
SAFETY = 0
LOVE = 40
BELONGING = 40
ACHIEVEMENT = 60
RECOGNITION = 60
FULLFILMENT = 60
```

```
O = openess 0
C = conscientiousness 0
E = extraversion 75
A = agreableness 85
N = neuroticism 60
```

Figure 18: Needs values file and Traits of personality values file

2.4 Three Proposals for Personality Models

In this section we present our three different approaches to the personality model: 'probabilistic model', 'needs based model' and 'amygdale model'. They were

developed by us as different ideas for our future design to overcome the combinational explosion encountered in previous nPME model, so the basics of that model are behind our three approaches. The chosen model for our future design is the 'needs based model' with some changes based on amygdale approach. Anyway, in this section we provided our three ideas for reference purposes.

In this section we present three different approaches to the personality model. These approaches try to overcome the combinational explosion that happens in the previous model [3] when combining all components.

The personality models we expose here are simpler, more computationally efficient but a big effort has been done to keep the coherence of the previous model, and therefore to build a solid personality.

The first part shows some assumptions about the personality concepts that are going to be **common for the three next models**. If in the next explanations of each model there is not remark about some common concepts, it is assumed that they will be used exactly as explained in the common section. The second, third and fourth part studies each model in detail, explaining the overall idea and its particularities, providing handy graphical diagrams. All these models use the common concepts.

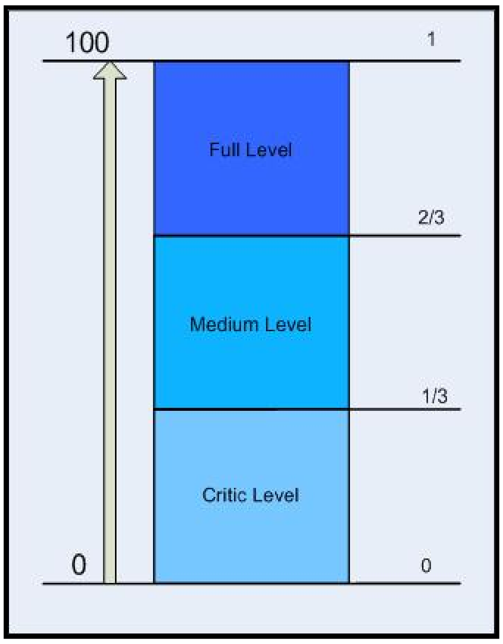
2.4.1 Common Concepts for the next three models

We based our approaches in the main components of the nPME model (needs, Personality, Mood and Emotions), combining in different ways to look for a design powerful enough to show behaviour and at the same time, suitable for implementation avoiding combinational explosion.

I. NEEDS

In Maslow's model there are five needs: *physiological, safety, love & belonging, self esteem* and *self actualization.* However, for our purposes we will consider only **the first three needs: physiological, safety and love & belonging**. The satisfaction of the needs is based in Maslow Pyramid [27], what means that need 1 must be satisfied until certain level before satisfying need 2.

Each need is based in a ***bucket* model**. This means that each need has a value from 0 to 100. Through time that value is decreasing in the same way a bucket loses water with a hole in its basis. When the bucket has only a small amount of water, it means that the robot must satisfy its need executing some actions to fill the bucket again. It is some kind of homeostatic approach: every need must have certain levels of satisfaction. For reasoning process the bucket is divided in three sections meaning three priorities when it must to be filled:



Full level [2/3-1]: it is not necessary to continue satisfying this need. Continue with higher needs.
Medium level [1/3 – 2/3): this need must be satisfied but only when the one that is currently being satisfied has finished. So basic ones will have higher priorities than superior ones in the same level, but only when the current one finishes its own satisfaction.
Critic level [0-1/3): Independently of which need is being satisfied, the need with critic level will immediately be satisfied. In case of conflict (several needs in the critic level), the most basic need will be satisfied first.

Figure 19: Need Bucket Model

As a general idea, needs are treated as buckets that are going empty basically through time and must be filled when necessary. That means that the robot must perform a set of actions to satisfy a need. If that performance successes, the respective need value will be increased, otherwise that value will be decreased.

For example, if the robot needs to recharge its batteries and it finds the recharging base, the need of battery will be reduced (i.e. the value of that need will increase leaving the critical level). If it cannot find its base, the need will continue decrease its value until the robot switches off.

II. PERSONALITY

Personality is based in five different traits (OCEAN model) that can take a value from 0 to 100 for each one. These traits give much versatility to give continuity to an individual's behaviour in different situations. However they have the problem that with those values they can define 100^5 slightly different personalities.

To avoid that combinational explosion cause by the traits, we will **create until four fixed personalities**. Although this seems to be very simple, the behaviour of the personality model becomes complex enough when adding the rest of the components.

The personalities proposed are different enough to make the witness perceive difference between one and another. We have given them a name that tries to be descriptive enough for the reader to imagine its ideal behaviour: ***friendly, unfriendly, aggressive*** and ***disobedient***.

How can this personalities influence the behaviour? In certain moments the system has to choose one execution way. In general, personality will give more priority to some of these ways, decreasing the chance of the rest for being chosen. Anyway, the proposed models treat on a different manner the inclusion of the personalities in the final model.

III. MOOD

Mood is going to be use as a thermometer of the intensity of emotions and the velocity of the actions. **It depends only of previous mood and the executed actions.**

Mood starts with some certain value, and **it changes continuously depending on the set of actions (script) that are executed**. So if those actions implies that the robot is satisfying one need, the mood will get better (higher positive value), and if the script is executed because it could not satisfy one need (*he wants to play and he does not find the ball)*, the mood will get worse.

The model of mood proposed by Lang [26] with two axes (*valence* and *arousal*) has been simplified in only one axe meaning the intensity of the mood (arousal) and the valence could be translated in the value of the arousal. If it is higher than 0, it means positive valence, and otherwise, negative valence. The higher is the value, the more intense is the mood.

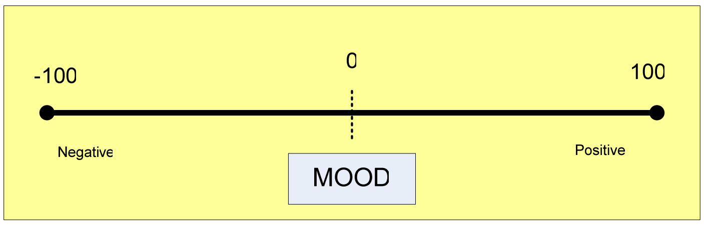


Figure 20: Mood scheme

IV. EMOTIONS

There are 6 possible emotions that can be shown by the robot, with different intensities: ***sadness, joy, surprise, disgust, fear*** and ***anger***. The moment each emotion is shown as well as its intensity **depends on the chosen model**.

V. SCRIPTS

A script is not a concept related to personality. However this concept is very important to explain next models. In our case, **a script is only a set of ordered actions that will be executed by the robot sequentially**. Those actions are grouped in a set because it is easier to create several scripts for each certain situation, instead of writing every action separately each time.

For example, script "**Wake up**":

1. Switch on leds;
2. Stretch;
3. Sit down;
4. Yawn;
5. Stand up; etc.

2.4.2 Probabilistic Approach

This model works with probabilities (some fixed and some that changes through time) to choose scripts and emotions. Each component of the nPME model and also each event have a list with probabilities for each script. This means for every situation, a script will be chosen when the product of all those probabilities gives it the highest value. This works in the same way to choose an emotion together with the script. The relation between the different components is depicted in Figure 21.

Every time the system updates needs (as explaining in common concepts), these values are recalculated with the ones proceeding from new events and previous emotions. Then the system will give an output list with the scripts ordered by the mean of the values they received from the rest of parameters, and the script with has higher probability will be executed.

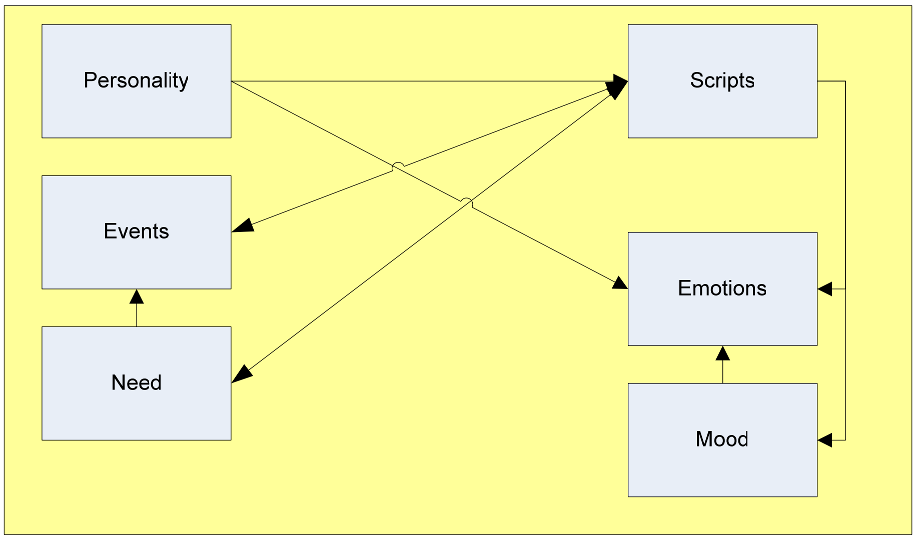


Figure 21: Scheme of the relation between parameters.[8]

For example, according to Figure 21, to choose one emotion, the system will do the mean between the probabilities that Personality, Scripts and Mood have chosen for every emotion. And the one that has the highest mean will be shown by the robot.

I. Particularities in nPME components

i) Personality

Each type of predefined personalities establishes different fixed probabilities for each script and for each emotion.

[8] The peak of the arrow means that this component changes its values according the component of the root of the arrow

E.g.: An aggressive dog will choose attack when he meets an unknown AIBO with higher probability than to flee. And also, an aggressive dog will have more probability of being angry than feeling joy.

ii) Events

Each event will have a predefined probability for each script. But that probability will only be used when that event occurs. When an event does not consider one script, it means that the probability of executing that script when that event occurs is zero.

Different events can happen at the same time, and depending on the needs and previous scripts some of them will be useful, and the rest will be discarded. This means that every event that occurs will be evaluated, but if because of the needs that event is not important, likely the power of this event for the overall election of the script will be minimized.

iii) Needs

Every script will be related at least to one need. Therefore each need will establish a fixed probability for each script. Of course the probability will not be used until a certain need is chosen, but when this happens, needs condition strongly the chosen of that script (or set of scripts).

The relation of the needs with the events will be real only in few occasions. One of them can be when the robot is starving and looking for the recharging-base. In those cases, a need will help the decision of the chosen script giving to that event a higher weight in the overall probabilities for the scripts. I.e.: the event can have double power in the value of the probability for that script.

iv) Emotions

The probability for an emotion to be chosen is the mean of the probabilities given by the script which is going to be executed next (at the same time the emotion is shown), the current mood, and the personality type for the AIBO.

II. Design Draft

The script that will be finally chosen is the one that have the highest probability, calculated as the mean of: personality, events and needs. The overall design of the this model is depicted in Figure 22.

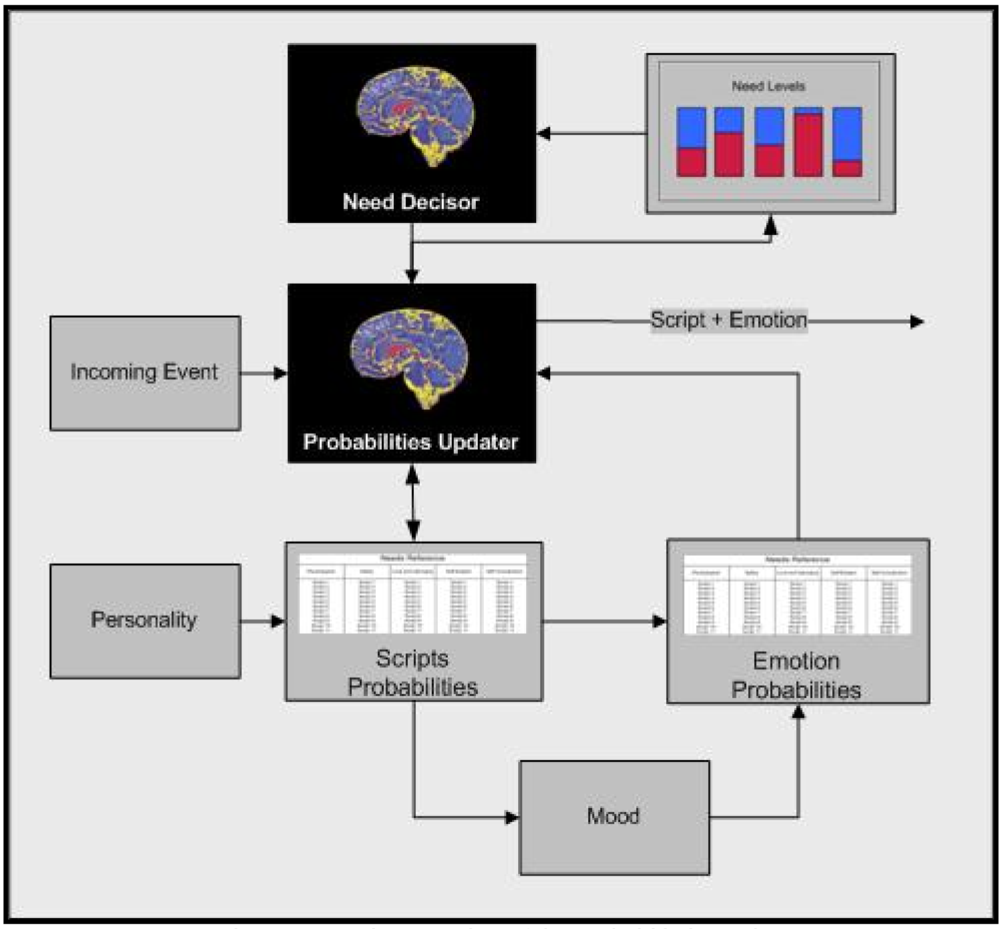


Figure 22: Design overview of the Probabilistic model.

2.4.3 Need Based Approach

In this model, the force that motivates the dog behaviour is the quest for the satisfaction of its needs. Each need will have certain combinations of scripts. To choose one or another will be driven by the rest of parameters of the model (events, personality, etc).

Basically, the system checks the different level of needs. According to the priorities and the level of the needs, the system choose one need to satisfy, what means executing the scripts that belongs to that need.

I. *Particularities in nPME components*

i) Personality

In some cases when AIBO try to satisfy one need, AIBO will have different ways to satisfy it (different script to choose in one transition). Only in those cases, personality will be used to choose one or another script. These elections will be fixed from the beginning, defined for each type of personality. This reduces considerably the computational complexity.

ii) Needs

The system, during performance has only one goal: ***satisfaction of needs*** and this is the main force of the system. One example of the 3 needs in one moment of the execution is shown in Figure 23.

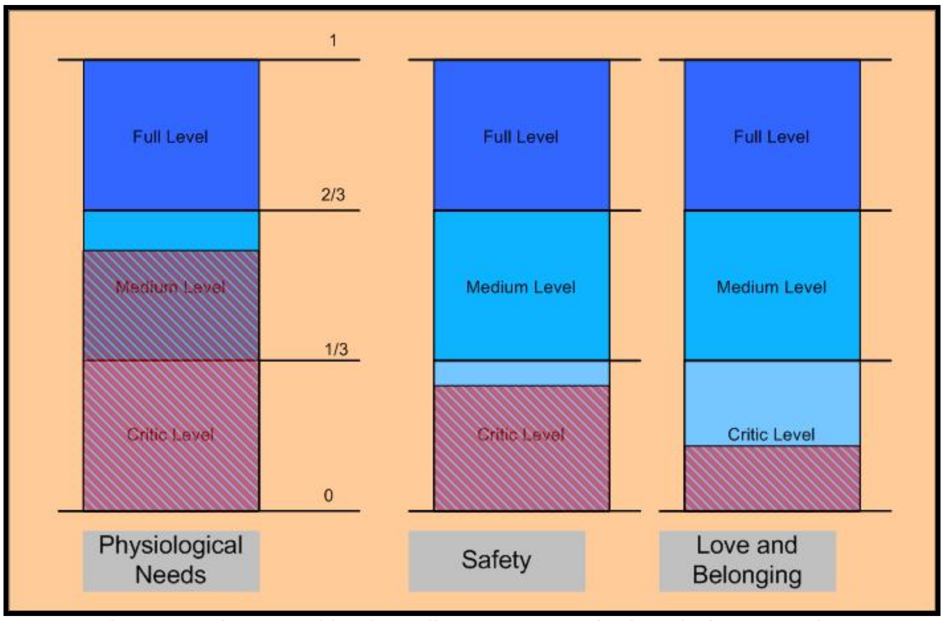


Figure 23: Three need-buckets diagram on certain time during execution

In the example of Figure 23 the lowest need level is for 'Love and Belonging', but since 'Safety' it is also in a critical level and its priority is higher, this will be the first need to be satisfied.

Each "bucket" has a different speed to empty. This speed depends on battery for the first one (physiological need) and exclusively on the time parameters for the rest. The lower priority for a need, the faster speed for being empty.

The speed for filling each one will depend only on the number of scripts that can be executed when that need is chosen as the most priority. Each script will have a different value in this process, according its own meaning.

iii) Events

Continuously the system will check both incoming events and level of the need buckets. The system will be always try to satisfy needs, and for doing that sometimes it needs some incoming events. In these cases, when a script requires a special event to be executed, AIBO will check whether that event has occurred. In that case it executes one script. Otherwise he would execute a different script.

For example, when AIBO is tired he will need to perceive the battery charger to know that he can fill his batteries. Depending on if he perceives it or not, he will recover energy (fill his first bucket) or not (leave it with same level) what will make him have different reactions both in mood and in actions.

iv) Emotions

The emotion is this model is totally fixed with the script. This simplifies the implementation and it does not decrease much the realism of the emotions. *For example, the script called 'Hear an Unknown Human' will be always associated with 'Surprise'.* More scripts have to be done to make sure that appropriate emotions fit with them. The intensity of each emotion will be controlled by Mood value.

v) Mood

What varies from time to time is the intensity of each emotion. The intensity will be controlled by mood. A positive mood always will mean positive emotions. And a negative mood will be related to negative emotions.

Each script will add or subtract a value for the mood depending of the meaning of the script. With the resulting value the intensity of the emotion will be calculated.

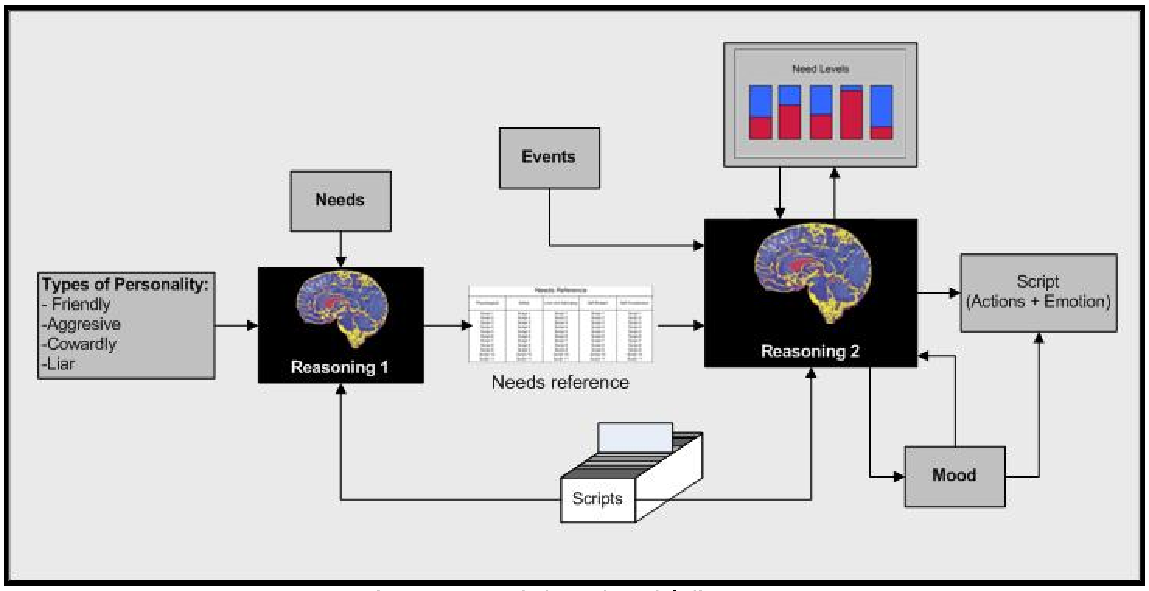


Figure 24: Needs based model diagram

II. *Design Draft*

First Inference Engine simply gives as output the list of available needs for a certain personality. It is like a filter. It is show in Figure 24.

The second one will choose the right script according with the priorities of needs, their levels, and the priorities of incoming events. Mood is updated from previous value also thanks to the chosen script, and that value will set the intensity of the emotion

2.4.4 Amygdale Approach

Since evolution is the most original invention and also the most useful in terms or survival, we will take a simple but powerful idea from nature.

Our brain has an original system to make us react to certain impulses. For survival reasons we have an 'old' part of the brain called ***amygdale***. This manages the affective activities. Besides it gives the response when we are hungry (it makes us look for food) and when we are in the presence of danger. In these cases, human survival requires a basic, fast impulse.

The 'newest' part of the brain is the ***neocortex***. It is involved in higher functions as sensory perception, generation of motor commands, planning skills, conscious thought, and language. This part is also the responsible of filter basic impulses through learnt social behaviours. *For example: if a person insults us, the first impulse might be aggressive, but according with the rest of information about the current situation (we may stay in the office) the best choice is to be calm, and response in another way.*

Basically the proposed idea is a robot driven by needs, but with a different structure of the brain that let him satisfy basic needs (physiological and safety needs) faster than the rest, and at the same time he can give a fast emotional response.

I. *Particularities in nPME components*

i) Personality

In this model Personality will only take importance in the satisfaction of the high-level needs (in our case, only need 3).

ii) Needs

As in the previous model, the schema of scripts classified by needs will be used. However, the approach to them is slightly different. The difference is that now we take into account the need of fast response in some cases (need 1 and 2). This is the reason that we have separate the basic needs (physiological and safety needs) that must be faster sometimes from the rest.

iii) Emotions

Normally emotions will be provided by each script. However, in this model there are some that are instantaneous and they motivate a change in the needs.

II. *Design Draft*

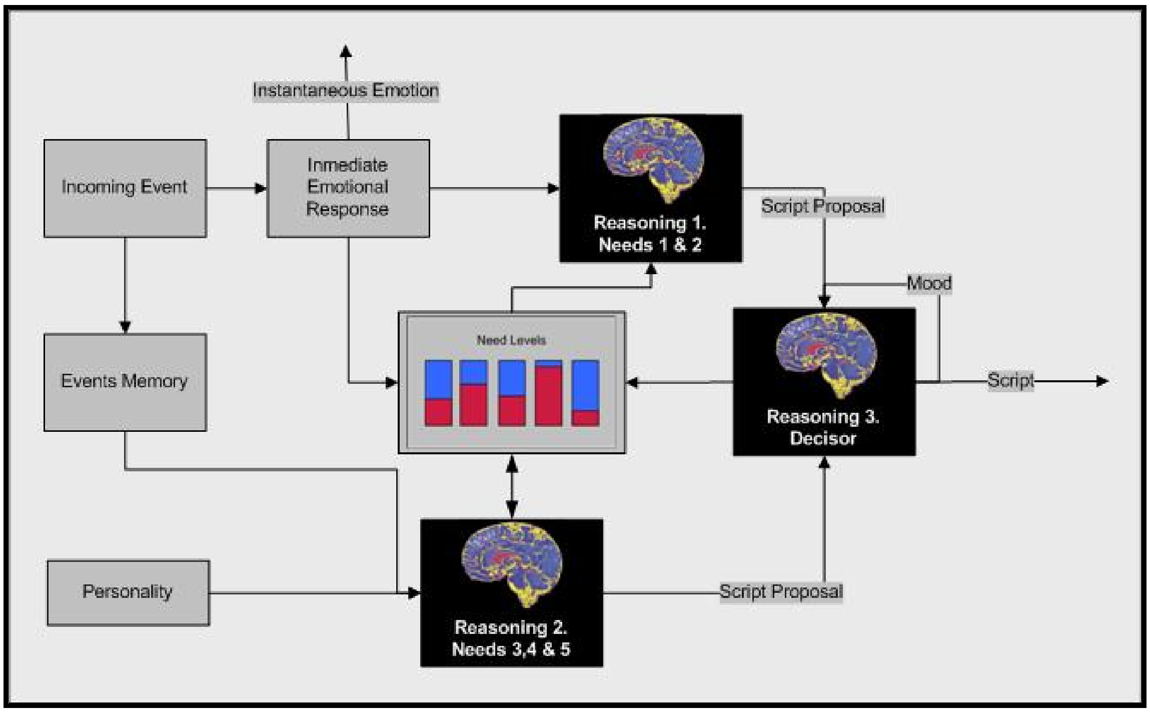


Figure 25: Amygdale based model diagram

In Figure 25 we show the idea of the design. There, the system gets an event. Depending on it (there will be a list with some special events), it will be processed for the module called "*Immediate Emotional Response*". This will give an instantaneous emotion and will modify in "needs level" the needs 1 or 2 (the satisfaction of these have suddenly been decreased). The consequence of this is that the system will try to recover the balance as fast as possible, so the Reasoning 1 will reason to propose a set of scripts to increase the level of basic needs.

The rest of the system is working in an analogous way such as the first model, and when a level of a need is low, the system will try to satisfy, proposing other script.

Then the *decisor* will get two different proposals of scripts: the urgent one or a normal one, and if the robot needs a fast response the *decisor* will choose the script proposed by the first inference engine.

If there is not a special incoming event, the system will work as usual, and sometimes there will not be scripts proposed by first reasoning. So the *decisor* will choose the script proposed by the second reasoning.

2.5 Technical Background

In this section we explain some important concepts for the whole understanding of the thesis, and how those components are related to each other. Some of the

concepts we will explain here are: robot, agent, structural programming, artificial intelligence, knowledge engineering, expert systems…

Our purpose is to create a continuous behaviour in a robot making it performance as a real companion dog as it has an alive personality. But what is a robot? Which tools do we need to implement its behaviour? Can we model a Personality with every programming language? In this section all these questions and more will be answered.

2.5.1 Artificial Agents

According to [35] a ***robot is an artificial agent, active, whose environment is the physical world***. First of all, an ***agent*** is, using Wooldrige definition, an "encapsulated computer system, situated in some environment, and capable of flexible autonomous action in that environment in order to meet its design objective". ***Active*** means that it is able to performance actions. ***Physical world*** means that its actions do not only happen inside a computer but also it interacts with real objects, and people. Combining these concepts we can see clearly that our AIBO robot is and agent that is going to act in the real world using a computer system to provide him the autonomy needed to let it be active choosing the actions to perform.

Focusing on the world, it has some features that are useful to mention to understand the complexity of creating a good robot [35]:

- Real world is *inaccessible*: sensors are not perfect and they can only perceive events when they are near the agent.
- Real world is *not deterministic* (at least from the robot point of view): there are many things than can happen out of robot knowledge (batteries can run out, connection can be lost, etc)
- Real world is *not episodic*: the effects caused by robot actions change during time.
- Real world is *dynamic*: so the robot must know when it can wait or when it has to react quickly.
- Real world is *continuous*: because states and actions came from a continuous set of physical configurations and movements.

The features of the real world always make the task of creating and implementing a robot be much complex than expected. Specially in our case in which the robot should be able to act in different changing environments (different rooms in a domestic house) with different human agents (babies, children, adults, other dogs, etc). There is a special type of agents modelled also by Wooldrigde that are specially suitable for our purposes because they have into account the information provided by the environment to performance one action or another.

2.5.2 BDI Agents

Wooldrigde set that an rational intelligent agent must have four characteristics. They have to be embedded in their environment, they have to be directed by goals to achieve, they have to react to changes in the environment and they have to be social in the sense that they can be able to communicate with other agents including humans.

So they proposed what it is called BDI agents where BDI means Belief, Desire and Intention.

- **Beliefs:** they represent the knowledge of the agent about the world. It can include inference rules that can lead to new knowledge about the environment. It is important to mention the use of the world belief meaning that the knowledge of the agent has may not be true.

- **Desires:** basically they are the goals that motivate the behaviour of the agent. The use of word desires indicates that those goals cannot be consistent, that is, two goals that are desirable but are contradictory.

- **Intentions:** they represent what the agent has chosen to do. They are desires that the agent has committed to certain extent.

Another important concept is the **plan.** Plans are sequence of actions that the agent can perform to achieve one or more of its intentions. Plans may be composed by other plans.

One of the first implementations of a BDI model, that later on has been repeated in different systems, is the one proposed by Georgeff and Ingran in the system called *Procedural Reasoning System* (PRS) shown in the Figure 26. The information obtained by sensors modifies the beliefs, which will be used to inference new plans trying to achieve some goals. The goals can change over time according to the external information. [28]

2.5.3 Hardware Platform

About the hardware construction, this is, the physical robot with its sensors and actuators (mechanical components), it is not our task. As mentioned in the first part of this chapter, we will use a commercial robot called ***AIBO***, made by Sony that has the advantage of being programmable. For more information look in sections 2.1 and 2.2.

The reason we chose this platform is because its price is low enough to be acquired by the university, it has enough actuators to provide some kind of natural movements and expressions, and it can be programmed as well as by Sony tools, by universal platforms where many users share their experience.

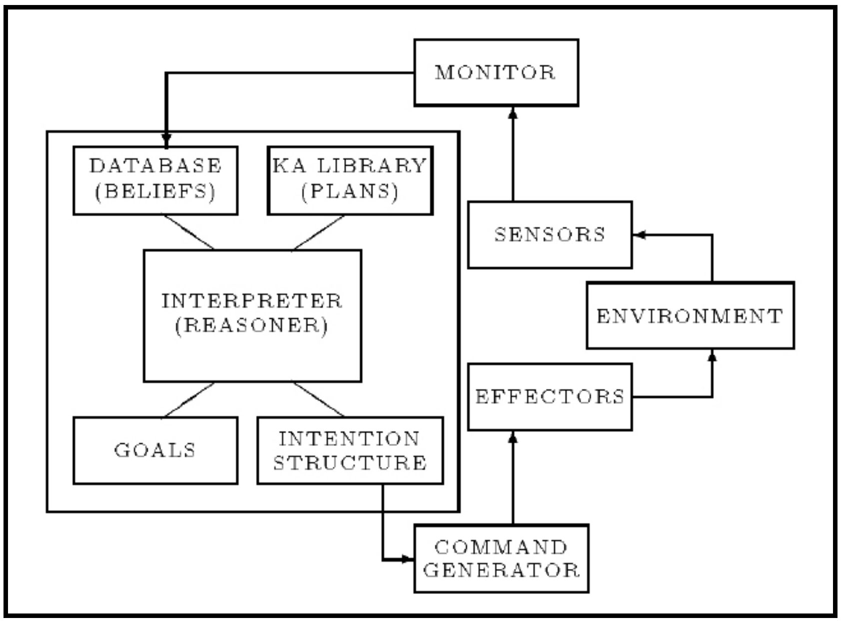


Figure 26: Procedural Reasoning System

2.5.4 Software basics

The fact that the robot is programmable means that using only software tools and writing the right commands, we can get an AIBO ready to perform in the real world (with more or less success). For that issue we can say that we need three different parts for a system like this, each one with its own specific techniques and languages:

Physical control: to translate the written commands into electric impulses to move AIBO joints; and also to translate electric impulses coming from its sensors into written words.

Reasoning control: to get the information already translated from the sensors and reason with it to understand the context where it is involved and in a logical way extract a conclusion to choose which actions the system must perform.
Overall system control: to join previous controls and make a correct exchange of information between them. It also gives the possibility to create a graphical interface in the computer to monitor every detail of the rest of the system and even interact with it.

About software tools we will focus in programming languages, artificial languages that are used to control machines, especially computers [32]. That is exactly what we need. But which computer language? Even although the classification is not accepted by everybody, normally it is said that there are 5 generations of programming languages [29, 32]:

- ***1GL*** or ***first-generation*** language is *machine language* or the level of instructions and data that the processor can work with (combination of 0's and 1's).
- ***2GL** or **second-generation*** language is *assembler language.* It is more symbolic what makes it more legible for the programmer. Then, the assembler also converts the assembler language statements into machine language.

- ***3GL** or **third-generation*** language is a "high-level" programming language, where symbolic actions are substituted by codes that are independent from the machine, more similar to human language or mathematics.
- ***4GL** or **fourth-generation*** language is designed to be closer to natural language than a 3GL language. Besides it uses some pre-made pieces to complete the programs. Some people identify this group with object-oriented programming, while another people think that databases languages belongs to this group.
- ***5GL** or **fifth-generation*** language is sometimes used to refer languages used in artificial intelligence, although this term is no much longer used.

The language that is used to control the AIBO robot is an operating system called Aperios, but nowadays there are some tools that allow the programmer to use more high-level languages, using *structured programming* or *object oriented programming*:

Structured Programming: there is only one entrance point and one exit point, although there can be different ways to join those two points. All the instructions can be executables without infinite loops, and normally the strategy is top-down, what means that the program is decomposed in stages or hierarchical structures. Normally a complex problem is divided in smaller problems that can be solved more easily.

Object Oriented Programming (OOP): it is faster than the previous one because it can recycle chunks of code called objects. Objects comprise their characteristics called attributes and the operations that are available to execute over them.

2.5.5 Robot software control

Both structured programming and object oriented programming can be used for the *physical control* and the *overall control system.* Structured programming will be better for the *physical* control since we only want to send commands created by the other systems and get its response. However, for *overall control,* OOP is more suitable since there are some concepts of personality (nPME model) that can be defined as objects, with their own executable code inside, what makes the implementation more robust and easier to understand.

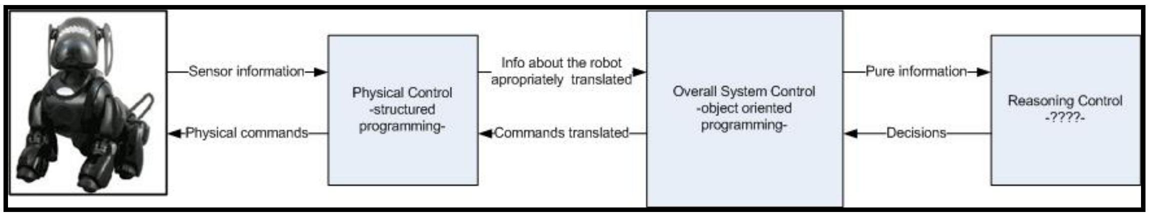

Sensor information
Physical commands
Physical Control -structured programming-
Info about the robot apropriately translated
Commands translated
Overall System Control -object oriented programming-
Pure information
Decisions
Reasoning Control -????-

Figure 27: Programming languages scheme

In Figure 27 it is shown the minimal overall scheme that a system to control a robot has. The robot gets some information about the environment with its sensors and that is translated by the Physical Control to make it suitable for the system to reason with it. The Overall System Control also translates that information into a language that the Reasoning Control can understand. The information that comes from the robot is only a part of the information of the world that the reasoning has to reason with.

So, how to characterize the world where the robot performances? We have seen that the real world has some features that make it very difficult to define in a suitable way for a processor to reason with it. Which is the information we have to use to characterize correctly this world? How can we process this information? How can we take decisions with it? Knowledge about the world is very complex and it is not complete, so it cannot be captured in mathematical structures. Therefore traditional programming techniques are not enough to treat it. However, there is a relative new discipline called ***Knowledge engineering*** that can help us providing some useful tools.

2.5.6 Knowledge Engineering

Knowledge engineering is related to artificial intelligence systems and tries to represent knowledge as well as human reasoning of a certain domain inside an artificial system. The task if a knowledge engineer is to extract the information from an expert and model it in such a way that allows some processing over it. There are some principles in knowledge engineering [32]:

- There are different types of knowledge just as different techniques and approaches to treat it.
- There are different types and expertise so methods have to be adapted to them.
- There are different ways to represent knowledge.
- There are different ways to use knowledge.

Thanks to this, it seems to be clear than traditional techniques of programming are not suitable enough for representing the complexity of the knowledge and reasoning with it. That is the reason that we use Knowledge engineering to create a knowledge-based system. That is simply, according to [30], a computer system programmed to imitate problem-solving in the same way a human does. It uses means of artificial intelligence and references to a database of knowledge of a particular subject.

Among the knowledge based systems, some of the most well-known are *expert systems*, *case-based reasoning systems* and *neural networks*. Briefly:

- ***Expert systems***: they represent expert knowledge as data or rules within the computer. They used a knowledge database with a wide knowledge about a certain domain, and this knowledge is only used when needed. If the knowledge changes, the logic of the system has not to be rebuilt [31].
- ***Case based reasoning systems***: they used solutions of past problems to find a solution for new similar problems. [32]
- ***Neural Networks***: it consists of interconnected processing elements called neurons that work together to produce an output function. It is failure tolerant and requires training to adapt the network to a certain problem. [32]

In our case we use Expert System approach because it is a very descriptive way of representing knowledge and allows for fast changes in the knowledge when the performance is wrong without the problem of modifying the logic of the system.

2.5.7 Expert Systems

Expert systems are a knowledge-based system. As we have seen, in traditional systems the user defines a new knowledge and he also has to define how this interacts with previous knowledge and he has to rebuild the sequence of processing instructions. However in KBS's, the user defines the new knowledge and the KBS integrates it in the existing application.

An expert system represents the information of a human expert in certain field of knowledge. But it works well with shallow knowledge, based on empirical and heuristic (rules of thumb or empirical knowledge gained from experience) knowledge instead of using deep knowledge that is based on structure, function and behaviour of objects.

An expert system is composed mainly of the next elements:

- **Knowledge Base or production memory:** it contains the main knowledge needed to solve problems. This knowledge is coded in the form our rules.
- **Working memory:** global database of facts used by the rules.
- **Inference engine:** basically it is an integrated set of problems resolution algorithms. It is coded and checked. It makes inferences by deciding which rules are satisfied by facts or objects, prioritizes the satisfied rules and executes the rule with the higher priority.
- **Agenda:** a prioritized list of rules created by the inference engine, whose patterns are satisfied by facts or objects in working memory.

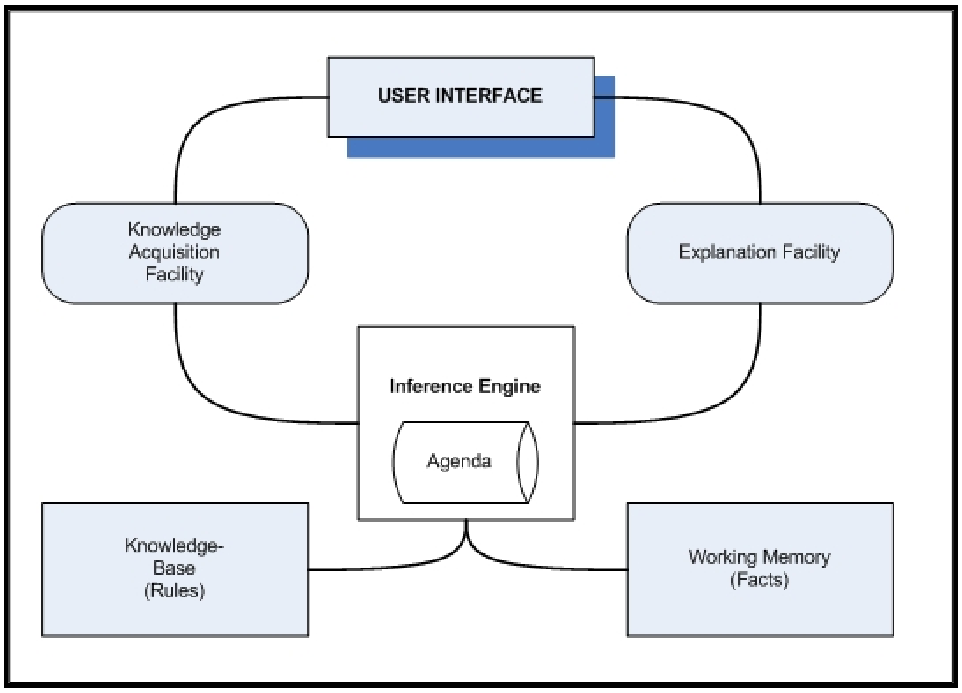


Figure 28: Expert System scheme

In our case, the expert system is going to be used to reason about the combination of external events and the internal emotional state based in nPME model. So this system will apply the appropriate rules to inference a set of actions based on the input data. Since it has to work in combination with other programming techniques, it is necessary to choose a language for the expert system that can exchange information with the rest of the components.

2.5.8 Programming AIBO

After seen different concepts in the technical background, it is the moment of putting all together to relate them with AIBO. Basically AIBO is a robotic platform that can be programmed and manage from a computer. So the goal is to make AIBO be an intelligent BDI agent able to recover information about the environment with its sensors, using that information to make a plan pursuing a desire and performance action by action of this plan to achieve a goal.

Since AIBO is a mechanical-electronic platform we need a programming language that allows us to use all its joints, leds, microphone and speakers. Moreover, once the physical control is achieved, we need another programming language able to performance some reasoning in the way an expert systems does. Besides, we also need another programming language for managing the physical control and the reasoning process as well as all the information messages between both and that provides us a graphical user interface.

In next chapter we will analyze all the information exposed until now to inference the final requirements of the system. These will lead us to take some decisions in the design of the system, in the hardware platform and also in the programming languages that fit best for our goals.

3 Analysis

Needs Based Companion Dog thesis comprises different theoretical disciplines: *man-machine interaction, technical subjects* (including both *hardware* and *software*) *and psychology*. They must be put together to achieve a solution of the problem definition mentioned in the first chapter.

This chapter makes an effort to analyze the different circumstances that surround the project to extract the main features that the system has to integrate. It uses the information provided in previous sections to deduce some conclusions for the future design of the system.

The chapter is divided in four main sections. The first three parts study the problem from three different points of view, according with the disciplines involved: use and interaction analysis, technical analysis and psychological analysis. The fourth section summarizes the ideas extracted from previous sections in the form of a list of requirements.

Chapter Overview:

3.1 Technical analysis

One of the reasons to start with this project was to make a suitable system for researching, what means that has to be easily understandable, easily modifiable and flexible enough to integrate new technologies in the future.

Since the system has to be suitable for researching, probably the most important point is the price. Although the robotic platform and the hardware costs cannot be easily reduced, among different options, we must choose a commercial robot that can be programmable.**[Req. 1]** It has to count with sensors to retrieve information from the real world and also an internal embedded processor to make it possible enough autonomy. Moreover different actuators are needed in a way that provides enough complexity for a behaviour like a real companion dog. **[Req. 2]**

Using appropriate software it is possible to safe more money. The key for this is the use of open-source tools, both programming languages and tools for the development. **[Req. 3]** These allow us the possibility of development software with cost zero under the protection of licenses like GNU public licenses for example. Another useful requirement is that the implemented code must be also open. This lets future users or researches study our code and change it what can give an additional value to these initials works. **[Req. 4]**

An effort has been done trying to keep in mind the simplicity of the system. That does not mean that the performance is not rich enough. The idea of simplicity is based on the fact that the tools that are going to be used are quite well known for the researching community. **[Req. 5]** Besides, the author will make a big effort to program things in an easy way when possible and also to keep abundant documentation in the manner of comments inside the code.

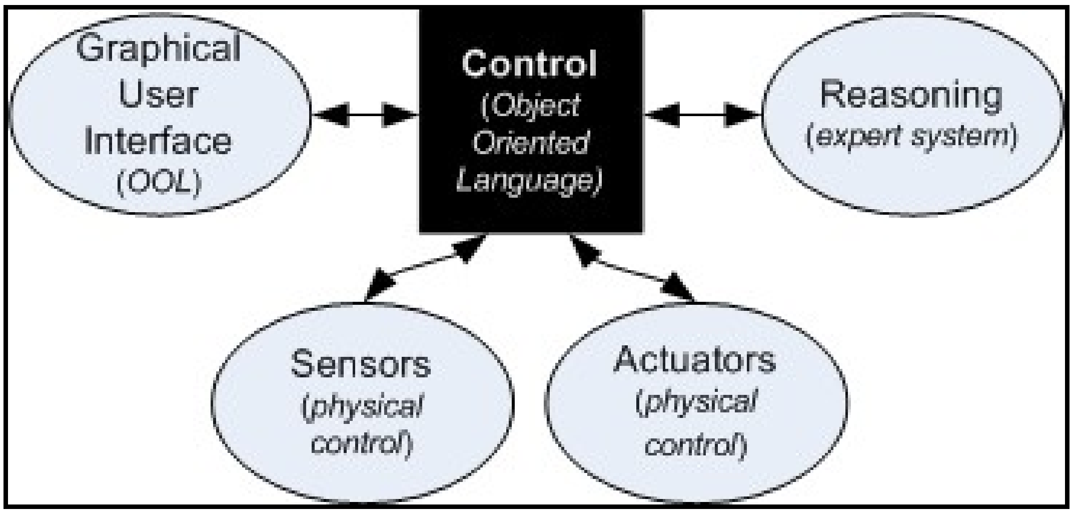
The system will be programmed using well known techniques of artificial intelligence to build complex agents. The BDI model will be taken into account behind the design as a seed for the overall project. Of course, the BDI paradigm is only a model, so according to the rest of our needs we will adapt that model to fit our goals. **[Req. 6]**

Since we want to develop a system able to retrieve and use some information from the world for its purposes, it is a knowledge base system. Inside this huge field of research, we chose a simple but powerful technique to develop the reasoning module: expert systems technique. This fact does not reduce the power and the performance of the system, but it does the complexity of the code. **[Req. 7]**

In this way of increasing the simplicity of the system, flexibility is also one of the main points for the design. With flexibility we mean the capability for the change of some pieces of the system, allowing the inclusion of new technologies or techniques (both the robotic platform and other software components). Therefore the system has to be as modularized as possible trying to increase the isolation between the different components. **[Req. 8]** Without doubts, this issue is very important for the future. Usually the highest cost in a software system appears during maintenance. So if we make a system able to adapt in an easy way to new technologies, and capable of substitute one of its modules, we help to overcome one of the most important handicaps for commercial systems.

The robotic platform has to performance in real world. That means that it has to interact with the environment. Therefore the system has to be able to get information from the world, process it and react appropriately always in real-time. **[Req. 9]** To let this take place, different processes have to run at the same time (recover information from the world while performing some behaviour, for example). This gives the idea of concurrent programming: it allows several processes to run in parallel using different information, but at the same time. **[Req. 10]** Thanks to that a continuous natural performance may be possible.

Since the whole system comprises different areas like hardware managing, reasoning tasks, remote control via user graphical interface (specially for testing), we have to find a software platform that let us to embedded those different techniques in one consistent program. So it is very important to use a programming language that allows us to work with other languages from it, as overall control. **[Req. 11]**


Graphical User Interface (OOL)
Control (Object Oriented Language)
Reasoning (expert system)
Sensors (physical control)
Actuators (physical control)
Figure 29: Components using different programming techniques

3.2 Use and interaction analysis

This project consists essentially in the creation of a companion dog for entertainment. Therefore, it has to be autonomous and designed for domestic use.

Entertainment means, according to Wikipedia [32], that the system must be designed to give pleasure or relaxation to the audience (one or more people). And the audience may participate in an either active or passive way. Therefore, to get some amusement with a robot, it can be useful to make it performance a wide variety of actions in combination with pleasant sounds and brilliant lights. **[Requirement 12]**

Since it has to be a companion dog, there is not going to be any specific task to accomplish like being a good shepherd, watchdog, etc. A companion dog provides only companion as a pet. So it has to perform same actions that an alive dog does: feeding, resting, doing some exercise, etc. **[Req. 13]** Moreover, normal dogs do not execute isolated actions. They combine many different individual actions to give a complex continuous behaviour. **[Req. 14]**

Companion dogs are usually called domestic dogs. This means that they have been adapted to human behaviour, living in human's houses and interacting whit them. In the case of the dogs, this relation is even narrower because many dogs live in continuous contact with children and babes. This implies that to achieve a good domestic dog, it is also necessary to make it easy to use, in the way that even a child should play with it without the need of previous instructions. **[Req. 15]** Besides, the interaction has to be as natural as possible, what means that in few hours, the behaviour of the dog has to be until certain point predictable. Otherwise, the children would be afraid if the robot shows fast unpredictable reactions. **[Req. 16]**

A normal dog is totally autonomous in the sense that it has the capability of thinking and acting independently by its own criteria. Most companion dogs are only dependent in the sense that is the owner who feeds them, and give some commands to execute. However, depending on the dog those commands will be obeyed or nor. So, it is necessary to create an autonomous behaviour, according to the circumstances that surrounds the system, giving it the chance to act even if there is not the presence of a human. **[Req. 17]**

3.3 Psychological analysis

The psychological model is one of the main parts of our system so a careful analysis of previous approaches is essential to succeed in our thesis.

This section will analyze the 5 personality models (2 already implemented and the 3 proposals) mentioned in chapter 2 to take out the most important advantages of each one for the future design.

3.3.1 Practical analysis of the behaviour developed by Sony

Unfortunately Sony seems unlikely to ever disclose the algorithms and models that represent the way AIBO reasons with emotions. However there is some data that can contribute to give an idea of the complexity of AIBO's behaviour. In the web page *Aibohack* [33] a comparison among every personality developed by Sony is given. According to that information, they conclude that much of AIBO's personality is a combination of a complex state machine, owner's interactions, the dog's mood, and a random number generator. To have an idea about the complexity of the system Sony created for AIBO ERS-7, next table is added:

Table 4: Sony AiboMind Complexities

ERS-7				
Personality	**States**	**Conditions**		
	#	**total**	**random**	**train**
AiboMind (core)	1435	6985	**281**	0
AiboMind 2(core)	2356	11558	**644**	0
AiboMind 3(core)	3731	19244	**1795**	0

The numbers of the table are only a rough indication of the complexity of the personalities, and where that complexity comes from. To interpret those values [33]:

- ***# of States***: reflects overall complexity of personality (more states usually means more features).
- ***Total number of Conditions***: It is the total number of conditions that will cause state transitions, so it also reflects the overall complexity.
- ***Random Conditions***: the number of the total that are influenced by a random number generator.
- ***Trainable conditions***: the number of random conditions that you can influence by training.

Moreover AiboMind personality for AIBO ERS – 7 uses a different learning technique and evolution approach than previous AIBO, so the given numbers are only an approximation of the real complexity.

Despite those data there is not more information available about how the personality was built. To overcome this handicap, some work was done in Man Machine Interaction Group in TU Delft to find out the personality model behind the robot. Results of the experiments conducted were summarized in the paper: "Emotional AIBO" by Chi Hyun Angela Lee [34]. According to our experiments the following were speculated regarding AIBO's cognitive model:

The user interaction can be classified into three different categories: **positive** (encouragement), **negative** (scolding) and **neutra**l (commands). Each one shows different sets of behaviours that seem to be random displays instead of being related to the intensity of the feeling.Positive and negative interactions with user always trigger corresponding emotional responses in AIBO irrespective of its instinctive state.

> *For example even when AIBO is hungry and is in search for the charge station, or when it is feeling sleepy and is inactive, it will still react happily when stroked on the back.*

However, these responses and even its emotional state appear to be only momentary, that is, **AIBO does not seem to have an on-going emotional state**.

> *For instance, when AIBO was scolded by being tapped on the third back button, it showed its anger by flashing red LEDs. However, it soon approached the user and raised its paw to be stroked. It is frustration of only few minutes ago seem to have been soon forgotten.*

The **emotional behaviours expressed when there was no interaction** with the user, that is, the emotional behaviours linked to the instinctive states of AIBO, **seem to be**

displayed at random or are displayed at such low probabilistic figure that its patterns are difficult to recognise.

> *For example, AIBO would flash green LEDs for no apparent reason even when there is no positive interaction with the user. Further observation may be needed in order to determine the pattern of these behaviours, if there is a pattern. [34]*

According to those results we can give some conclusions for the incoming system:

First, instead of having only three categories of interaction, this must be richer according with the personality and the situation itself. Therefore different personalities will give different responses for same interactions. **[Req. 18]**

Second, it is more interesting if the emotional state is more continuous during time. There should be a parameter that increases and decreases the intensity of emotions in small amounts to let more natural transitions. **[Req. 19]**

Third, some random actions can introduce fun into the dog's behaviour. However, those actions have always to be related with some kind of internal state in the robot and the context where it is performing. **[Req. 20]**

3.3.2 Analysis of Personality Model for Companion Dog

In Personality Model for a Companion AIBO [3], the authors have developed a complex and flexible framework to characterize a personality model for an AIBO robot. That framework will be very useful for our goals as an inspirational source of knowledge. Moreover several parts of the code will be recycled for integration in our new system.

The developed system was a ***simple reflex system,*** what means that as soon as the system perceives some event from the environment, it acts performing some action [35]. It combines for the action election an emotional internal state with the perceived event. But the key is that every time there is an event, there is also an immediately response: no matter how often this event occurs, the response will be always the same.

In the new system of 'Needs Based Companion Dog', instead of a simple reflex system, a ***system that keeps track from the world*** [35] is proposed. **[Req. 21]** The difference is that the proposed system will have some idea of the consequences of the actions it will perform. *E.g.: If AIBO's owner goes into home, and AIBO sees him, it will try to play with the owner. AIBO will know that after that action the owner can decide to play or not, and no other situation is possible. So depending on owner's reaction, AIBO will change its emotional state and performs one set of actions or another.* The fact is that with the extension, AIBO's behaviour will be increased providing more natural responses, because future actions will depend directly on past events, emotions and actions. So if the same event occurs several times, the robot will show different behaviours according to internal variables and data stored in memories.

In Personality Model for a Companion AIBO, many actions have been implemented. Indeed there are more than which are actually used, so these actions will be used in our system. **[Req. 22]**

Looking at the 'brain' in Figure 30, and according to the personality components explaining in section 2.3.2.1, we can realize that a combinational explosion appears when trying to combine so many parameters. This is of course a serious problem for the implementation. So in the new model this combinational explosion has to be reduced drastically but keeping enough complexity to show a rich behaviour. **[Req. 23]**

The first step to remove that complexity is the elimination of *Goals, Preferences* and *Standards*. They introduce more problems for the implementation than benefits for the behaviour.

Our model was planned initially as an extension of the system [3]. A more deep study showed that the system was not suitable for our purposes. However there are some parts and ideas that continue being useful for us, so for our system we will use same programming languages than in [3]. This will let us some reuse of code even if we have to start from the implementation from scratch.

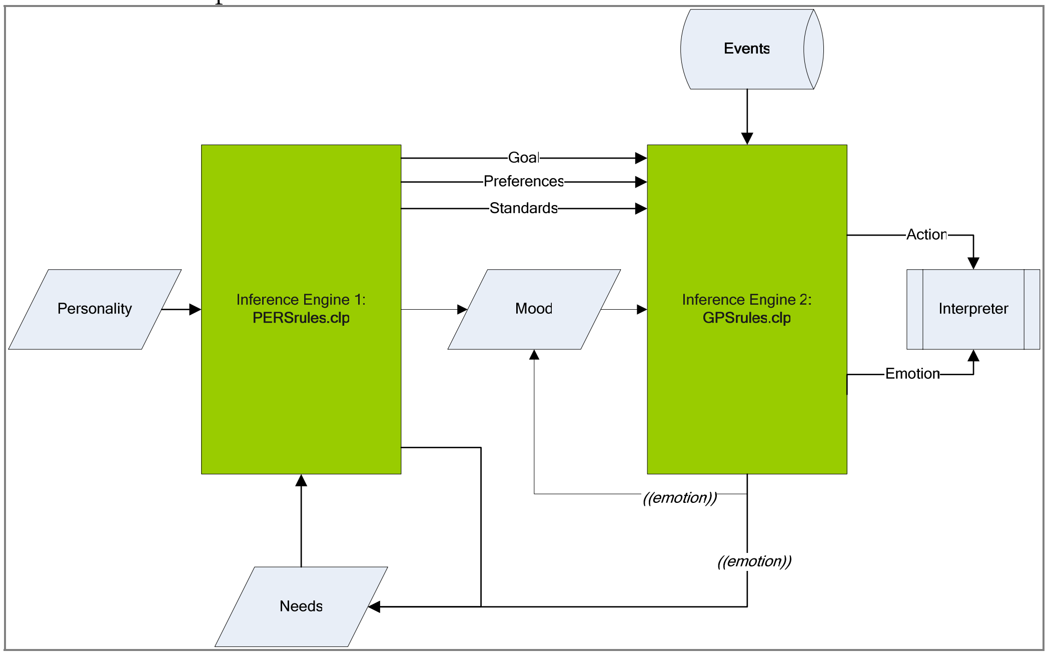


Figure 30: Reasoning Scheme in Personality Model for Companion AIBO

3.3.3 Analysis of the three proposals: Probabilistic, Needs and Amygdale approaches

After analyzing the three different models that were proposed in section 2.4, we can conclude some ideas about the personality model that will be the seed for the final model designed in this thesis.

About the common concepts for the three models (*personality, needs* and *mood*), we will keep those ideas for the future design:

- Four different personalities will be developed. Each one will condition ways of execution in certain moments during performance. **[Req. 24]**
- Needs will be treated as buckets that empty during time. They are filled when some actions to satisfy a need are taken. **[Req. 25]**
- Mood will be treated as one axe meaning the positive or negative valence according to the sign of the value, and the intensity is wrapped by the absolute value. **[Req. 26]**
- Scripts will be used as a container of sequences of actions with some coherence between them in appropriate situations. **[Req. 27]**

About the ***probabilistic model*** the idea behind it is very powerful. It gives us the chance to create many different behaviours. But it has a serious problem: it gives another combinational explosion, because each personality component will have different probability values for each script. This implies a big problem for adjusting the behaviours (to make them be coherent) and for modifying every simple detail. So the maintenance of this system becomes terribly difficult. That is the reason that we rejected this model.

The ***needs based model*** seems to be the best one to implement. It is simple enough to let a solid design and implementation and it also gives enough potential to create very complex behaviours. So the new design will be based on the satisfaction of the first three needs of the Maslow's model. **[Req. 28]**

About the ***amygdale based model*** it is very similar to the *needs based* one, but its design is much complex for almost the same results. We also rejected this model. However there is one nice idea in this model which will remain in our design: there are some incoming events that can modify the needs before one need is chosen for satisfaction. *For example, if one unknown person enters home, the safety need of the robot should be affected when perceiving that.* So we will keep this useful feature that gives more coherence to the robot's behaviour according to the context. **[Req. 29]**

3.4 Conclusion

The three previous analyses give the requirements our project must accomplish.

After analyzing the different aspects of the project, a list of requirements has been reasoned trough the different sections. In this one we summarize all those in a simple list to let the reader has a wide overview of the requirements our system must fulfil. These are shown in Table 5.

Table 5: Requirements

#	Technical
1.	**Commercial programmable robotic platform:** popular programming platforms.
2.	**Robotic platform has to have enough sensors, embedded processor and actuators.**

3.	**Cheap for research:** open-source tools.
4.	**Open code:** open, easily understandable code.
5.	**BDI agent model as basis:** this will be the basis for design adapted to our particular case.
6.	**Popular programming tools:** use popular programming languages and tools to favour the ease modification.
7.	**Well-Known AI techniques:** reasoning made using expert systems.
8.	**Flexible system:** high modularization. Ease change of components.
9.	**Real-time system:** it has to react in an appropriate time to events in the world.
10.	**Concurrent programming:** several processes should be run at the same time.
11.	**General programming platform able to work with other technologies:** it has to control physical actuators and expert systems programming languages.
#	**Use & Interaction**
12.	**Entertainment robot**: introduce variety of actions, pleasant sounds and brilliant lights.
13.	**Companion robot:** behaviour similar to a normal companion pet.
14.	**Continuous behaviour:** the sequences of actions must be continuous without annoying stops between them.
15.	**Domestic robot:** ease of use. No special instructions required.
16.	**Suitable for children:** natural and predictable behaviour. No unpleasant actions.
17.	**Autonomous robot:** the robot should performance without a person's control.
#	**Psychological**
18.	**Personal behaviour:** personality defines different behaviours in same situations.
19.	**Continuous emotional state:** emotional states have to evolve slowly in the most of the cases.
20.	**Coherent reactions:** actions should be related to both internal state and environment.
21.	**World aware system:** it gets information from the external world that conditions its behaviour.
22.	**Reuse of actions:** many actions that were implemented in [3] will be reused.
23.	**Avoid combinational explosion:** keep the parameters of the system reduced enough to avoid that combinational explosion.
24.	**Fixed personalities:** there will be four different personalities.
25.	**Bucket needs:** needs are treated as buckets with water that empty with time. They can be filled when some actions satisfy the right need.
26.	**One axe mood:** the value of the mood will summarize valence and arousal.
27.	**Scripts:** scripts will contain sequence of actions for certain situations.
28.	**Needs based model:** the overview of the design will correspond to ***Needs Based Model***
29.	**Amygdale modification:** some events will affect the needs before the election of the need to satisfy.

4 Design

This chapter provides a logical evolution starting with the functionalities the system has to provide until the deduction of the final system design. Some diagrams will be also given to ease the comprehension of the proposed architecture.

Taking the requirements from previous chapter, first section will explain the functionalities that the system must provide. Then, each group of functionalities is explained in detail with the specific circumstances associated to them (sections 2-4). Thanks to the functionalities, in the fifth section, a very general diagram of the system design is shown. Later, in the sixth section, the main modules of the diagram are decomposed in a more detailed picture for better understanding, but without losing the global architecture. Some remarks of the components will also be added in this part, and also some constraints are taking into account to ease future implementation.

Chapter overview:

4.1 Functionalities

Using the requirements as a start point, the functionalities that the system must provide are inferred. These are explained in this section where some useful use-case diagrams are provided.

A functionality is a "the sum of any aspect of what a product, such as a software application or computing device, can do for a user". [36] This is normally wrapped by what is called User-Case Diagram in the UML notation [37], that is shown in the first part. Taking that perspective we go further in the development of the system and we divide the functionalities of the system in four different groups conceptually different, giving the requirements that each one satisfies.

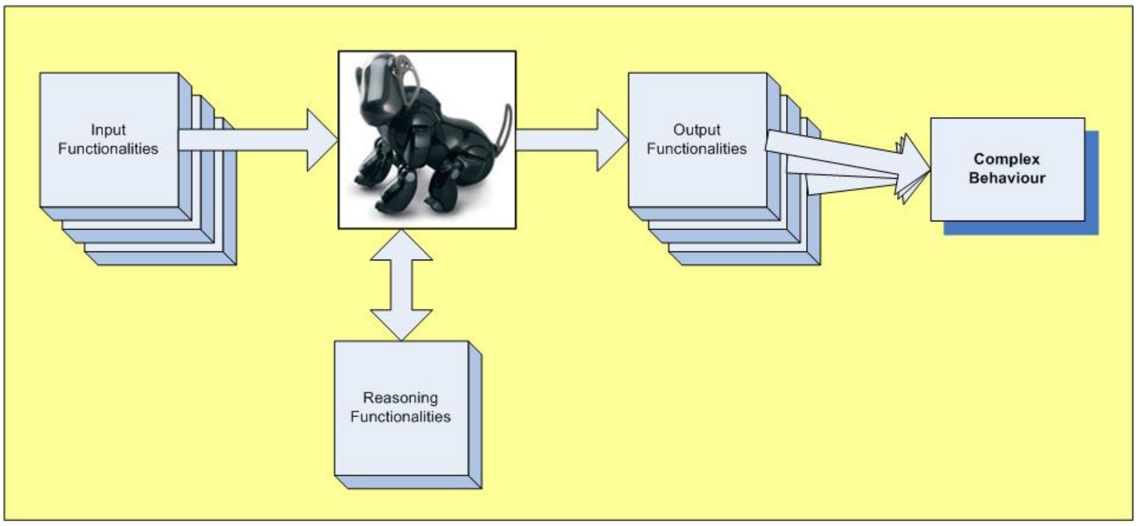


Figure 31: Functionalities Scheme

4.1.1 Use case diagram

The use case diagram shows the usage requirements for a system [38]. It represents a unit of interaction between a user (human or machine) and the system. The basic elements that a use case diagram has are [38]:

- **Use cases**. A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.
- **Actors**. An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures.
- **Associations**. Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case.

Figure 32 presents the User-case diagram. We depicted three different actors since they are the parts of the system that play an active role in the sense that they receive and/or create essential information for the performance of the whole system.

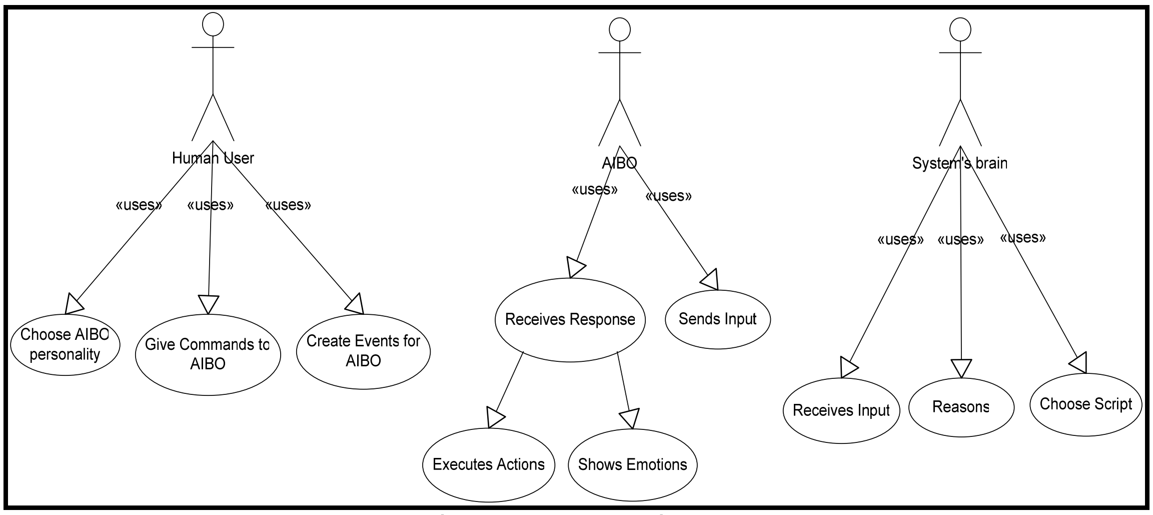


Figure 32: Use-Case Diagram

Next tables present the use cases. We have added what is called pre-conditions, post-conditions and exception conditions that provide a black box description of the use case. The pre-condition describes the state the system must be in before the use case starts. The post-condition describes the state the system must be in after the use case completes.

Table 6: User-Cases Diagram description

Title	**1. Choose Aibo Personality**
Pre-conditions	The system allows you to choose the personality.
Use case description	The human actor can choose the personality of its companion dog through the interaction with the system.
Post-conditions	Personality has been stored in the system.

Title	**2. Give Commands to AIBO**
Pre-conditions	The events GUI is visible
Use case description	The human actor can give direct commands to the AIBO in the sense of oral commands like a normal dog. In this system those commands are wrapped through some kind of graphical interface.
Post-conditions	The command event is stored into the fact base of the reasoning process to be used.
Exception conditions	That command does not exist in the GUI.

Title	**3. Create events for AIBO**
Pre-conditions	The events GUI is visible
Use case description	The human actor, using the graphical interface can create different events for AIBO. A command is an event, but also the image of the owner is one of them.
Post-conditions	The event is stored into the fact base of the reasoning process to be used.
Exception conditions	The event does not exist in the GUI.

Title	**4. Receives Response**
Pre-conditions	Some actions from an script are being sent to the AIBO robot.
Use case description	AIBO actor receives different commands for the system in a language that it can process.
Post-conditions	AIBO executes those actions (showing movement, lights or/and sound).

Title	**5. Sends Input**
Pre-conditions	AIBO sensors are working to retrieve information or/and AIBO processor is receiving commands to execute.
Use case description	AIBO actor uses its sensors to get information from the environment and translate this info in a language that the rest of the system can understand and process. It also gives feedback when a command coming from the system is executed. That also constitutes an input that AIBO sends to the rest of the system.
Post-conditions	The system gets this information and incorporates as facts in the reasoning process.
Exception conditions	Message is not treated as event and it is ignored.

Title	**6. Execute Actions**
Pre-conditions	Action has been sent by the system.
Use case description	AIBO actor performances different actions that have been previously requested by the system. These actions have a sequential order and most of the times they consist in complex actions as the result of combining simple continuous movements.
Post-conditions	Stop executing that action.

Title	**7. Shows Emotions**
Pre-conditions	Emotion expression has been set by the system.
Use case description	AIBO on request shows different emotions in the sense it switches on different colour-leds in its face at the time it performs some kind of actions. That results in the appearance of complex emotions.
Post-conditions	It continues with the leds on.
Exception conditions	Specific command to reset Leds.

Title	**8. Receives Input**
Pre-conditions	An event has been created as fact in the reasoning system.
Use case description	The reasoning actor receives different data from the rest of the system, consisting in external and internal events (all coming from AIBO actor) and internal emotional state of the system
Post-conditions	The event is store for future inference processes.

Title	**9. Reasons**
Pre-conditions	An event is stored as fact in the reasoning system and the control asks for inference process.
Use case description	The reasoning actor has to inference a specific need to satisfy through the evaluation of the received events and its own internal emotional state.
Post-conditions	The need is chosen.

Title	**10. Choose Script**
Pre-conditions	A need has been chosen.
Use case description	The final result of the reasoning system will be a script that is simply a sequential set of combinations between actions and emotions resulting in the appearance of a complex behaviour.
Post-conditions	The script is executed.
Exception conditions	There is a failure in the connection with the robot.

Taking this into account and according to the requirements described in previous chapter, we can deduce the functionalities that have to be available in our system. In a general manner, we can classify the functionalities in four different groups, according to the concept behind them. That helps us to accomplish the requirement number **14**. Moreover, each group will accomplish some of the overall set of the requirement, resulting in the satisfaction of all of them:

- ***Input functionalities***: to make the system get information from the environment.
- ***Output functionalities***: to perform simple actions and show simple emotions.
- ***Reasoning functionalities***: to combine stimuli and internal state to inference different situations where performance combinations of actions and emotions.
- ***Complex behaviour functionalities***: to show high-level behaviour combining sets of simple actions.

4.1.2 Input Functionalities

These functionalities provide the system with the information about the environment. There are two ways to do it: through translation of robot sensors or using a GUI where some events can be created. However, the system will treat that information in the same way, to let future improvements. In this manner, the input functionalities that the system can perform are:

1. Distance measure
2. Sound recognition
3. Vision recognition
4. Touch detection

Requirements addressed: **1, 2, 6, 11, 15, 17**.

4.1.3 Output Functionalities

These provide basic actions that the robot will perform to interact with the environment through robot actuators. Here, it is also included the emotions that the robot can show, since actuators also used for that.

- **Actions**: using its paws, head, mouth, neck, ears, tail and speaker.

Table 7: Basic actions

Walkahead	Walk back	Turn right	Turn left
Sit	Stand up	Lie down	Scratch
Dance	Give paw	Say hello	Bark
Move ears	Wag tail	Purr	Stretch

Example 1: moving its paws walk through its owner.
Example 2: bark moving its mouth and emitting barking sounds through the speaker.
Example 3: sitting using its paws and move head and mouth to salutate.

- **Emotions**: using its leds, head, mouth, ears, tail and speaker. These correspond directly with the emotional mode. Each emotion can have different intensities.

Table 8: Emotions

Sadness
Happiness
Surprise
Disgust
Anger
Fear

Example 1: move its head down and wag it to show sadness at the time a sad sound is emitting by the speaker.
Example 2: move up its head and its ears lighting some leds with the mouth open to show surprise.

Requirements addressed: **1, 2, 12, 18.**

4.1.4 Reasoning functionalities

These functionalities provide the way in which the information is managed from the input to the output.

1. *Translate inputs into incoming events.*
2. *Choose the need that has to be satisfied.*
3. *Choose the set of actions and emotions to perform.*
4. *Update Internal State*
5. *Translate actions and emotions into robot's language.*

Requirements addressed: **1, 2, 3, 4, 8, 9, 19, 20, 21, 22, 23, 24, 25.**

4.1.5 Complex Behaviour functionalities

To show a complex behaviour some basic actions in combination with emotions must be shown. Moreover they must be coherent with the context according to incoming events.

So the combination of simple output functionalities (actions + emotions) as the output of the reasoning taking into account the input events gives a complex behaviour that tries to be in coherence with the events that occur around the robot.

For example: if the robot needs to recharge its batteries and there is no image of the recharging base, it will start look for them until it reaches it.

Another example: if his owner comes home making some sounds and the robot missed him and can hear that sound, the robot will look for him to get some strokes. Otherwise, although the owner comes home, if the robot needs to recharge its batteries probably will ignore its owner.

Requirements addressed: **1, 2, 7, 10, 13, 14, 16, 24**.

4.2 Input Functionalities

According to the book "*Artificial Intelligence: a modern approach*" [35] an Agent is every system, that can perceive some stimuli from the environment with 'sensors' (eyes, camera, microphones, etc) and also can act in this environment with its effectors (hands, paws, mouth, mechanical tail, etc). In this sense AIBO robot can be considered an agent.

An artificial agent that is acting in the real world will only be able to perceive the stimulus that its sensors can recognize. It is the same with human beings that can only perceive five types of stimuli (one per sense). In fact a human is also an agent acting in our 'real world'. This limits the conception of the world that an agent can have, so for an agent that can only perceive sounds, the world is only composed by sounds and no matter silent objects, colours, smells and so on. Therefore, due to this perception limits, it is important to define 'an appropriate world' for each agent. This will only be defined by the events or stimuli that its sensors can recognize.

AIBO is an agent performing actions in our world, but the one it can perceive will be only based in the information recovered by its sensors. The most important sensors in AIBO are:

- **Sight**: 350.000-pixel CMOS image sensor.
- **Ear**: stereo microphones.
- **Sense of touch:** some touch sensors in head and in back.
- **Distance sensor:** two infrared distance sensor and an edge detection sensor.
- **Battery sensor**

All of these sensors should be used to process the world where AIBO lives and let it make some reasoning to know where it must go, where its owner is, etc. If we divide all the information it can recover from reality in small chunks, we have what we call '*events*'. Our system will work and reason with these events.

However we have one limitation: the information that the sensors get, especially camera and stereo microphones, must be pre-processed and translated into data that our system can understand. So we need techniques from *Visual Recognition* and *Sound Localization* and *Recognition*. Both are beyond the scope of this project. Besides, to let AIBO move in the world, we need some kind of *avoiding obstacles algorithm*, but for the moment it will not be implemented.

In Figure 33 there is a schematic representation of how AIBO can perceive and process the world surrounding it. As shown in the picture, every event will be treated by the 'brain' to make AIBO performance some behaviour.

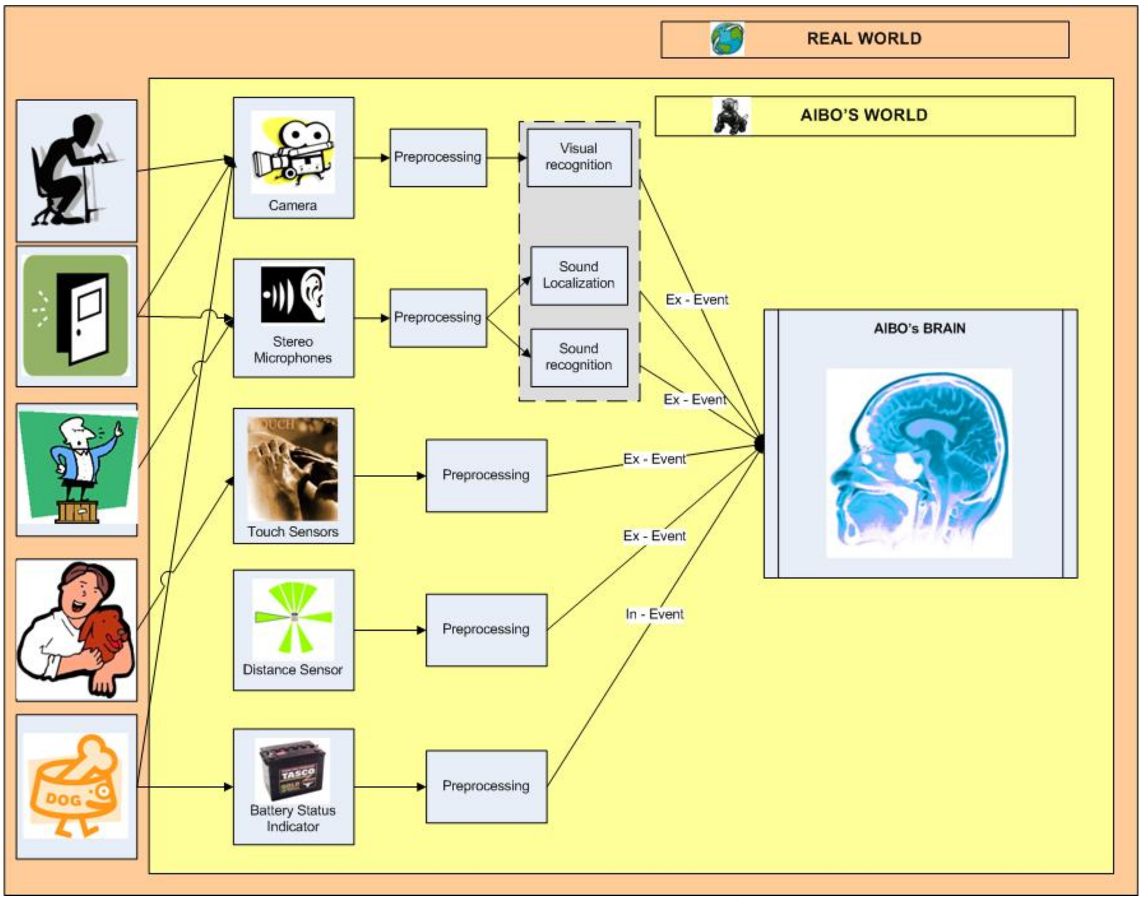


Figure 33: AIBO's world. Schematic Diagram

Also in the picture we can see a square surrounding the modules "Visual recognition", "Sound localization" and "Sound recognition". That square means that those modules will not be available in the current project.

4.2.1 Modelling the world

For testing purposes AIBO will move in a small area (4x2 meters) which simulates a normal house. AIBO's world is going to be composed of *agents, objects* and *events*. An *agent* will perceive and act in AIBO's world so they can perform actions and can react in the presence of AIBO. An *object* is an inert, motionless thing, unable of performing any kind of action (except make sounds), but it can condition AIBO's behaviour if it can be perceived. Basically AIBO only can perceive the image of an object.

Agents and objects will be perceived by AIBO's sensors in the form of 'events'. So an event can be defined as a type of information that is susceptible of being recognized by AIBO's sensors (*E.g.: an agent will be recognized by AIBO because of its image –eye sensor-, its sound or voice –microphone sensor- or even by its touch – touch sensors.*) These will be the key to trigger one or another behaviour.

The **agents** (both biological and mechanical) that can participate in a scenario are:

Table 9: Agents

Owner
Human Friend
Human Enemy
Unknown human
AIBO
Another AIBO: friend
Another AIBO: enemy
Another AIBO: unknown

There are some **objects** that can be also in a scenario. There can be many more objects, but for testing purposes only some will be used:

Table 10: Objects

Door
Ball
Bone
Food
AIBO's recharging base.

4.2.2 Events

Agents and objects will only be perceived by AIBO depending of their intrinsic characteristics: a ball mainly will be perceived as an image, but a person can be identified not only by image but also because his voice. So agents and objects must be translated into **events.** They are basically occurrences along time that can be perceived by AIBO sensors (ball image and owner image by AIBO camera, owner voice by AIBO microphones).

It is important to say that the most of the events than can occur will not be perceived directly by AIBO sensors because sound and image recognition techniques are beyond the scope of this project. But, since we want to test AIBO performance in a real world, events are necessary. So events will be wrapped as buttons in a graphical user interface (GUI). So when one event-button of the GUI is pressed, AIBO will perceive that certain event has occurred and it will act consequently. However a big effort in the system has been done to make possible a future substitution of that GUI component for real sound and image recognition components.

The **events** are described according to the sensors that get the information. In this way it will be easier to have every event well classified to implement it correctly. One important thing classifying these events is taking into account the agent that causes it, when there is one.

- ***Touch*** sensor***:*** shown in Table 11.

Table 11: Touch events

Owner touches back
Friend touches back
Enemy touches back
Unknown person touches back

- ***Camera:*** In this part maybe it could be logical to consider the position of the object, the size, etc. However it will not be necessary because using the GUI to create those events it can be supposed that every object will have the right size and will be in the right position that conceptually will always be exactly in front of AIBO.

Table 12: Video events

Image of a ball
Image of a bone
Image of owner
Image of friend
Image of enemy
Image of unknown person
Image of friend Aibo
Image of enemy Aibo
Image of unknown Aibo

- ***Microphones:*** the possible events that the microphone can detect are classified in Table 13 in three different groups: *sounds,* without words spoken, *commands* that AIBO must obey depending on the situation, and *characterization commands,* used to encourage AIBO in some behaviour or to reprehend it because of another one.

Table 13: Audio events

Sounds
Knock door
Door open sound
Door close sound
Bark sound from another Aibo
Commands (from the owner)
“Come here”
“Play”
“Take the bone”
“Look for the ball”
“Look for the bone”
“Let’s go to street”
Characterization commands
“Good Dog”
“Bad Dog”
“No”
“Yes”

4.3 Output Functionalities: basic actions

One of the basic features of a companion robotic dog is its ability to perform real actions in real world interacting with its environment. Here is when the output functionalities are necessary.

With output functionalities we mean a set of basic actions performed by AIBO actuators like shaking head, open the mouth move a paw. These simple movements can be combined in real time to show a more complex actions like sitting, laying, stretching, etc. As output functionalities we also include the function of playing sounds or the possibility of switching on some LEDS in the robot face. This provides richer combinations of actions that make AIBO able to show some emotional expressions.

As we saw in section 4.1.3, we have divided the output functionalities into “Actions” and “Emotions”. We are not mentioned again the actions that are available, but a remark has to be done: the division between actions and emotions has been done as an effort of clarity, because the border between them is diffuse sometimes. For example when showing happiness the robot will use some LEDS but also it will wag its tail, as real dog would do. Anyway all the simple actions can be performed alone without emotions, and also emotions can be programmed using only leds and sounds. However for realistic behaviour many times is better to mix some actions from these two groups.

Another important idea is that these actions, or these combinations of simple actions, by themselves do not show complex behaviour. It is more complex combination on time of those actions together with emotions what will make the robot appear real. That is what we called *complex behaviour* (that is explained in the section 4.5).

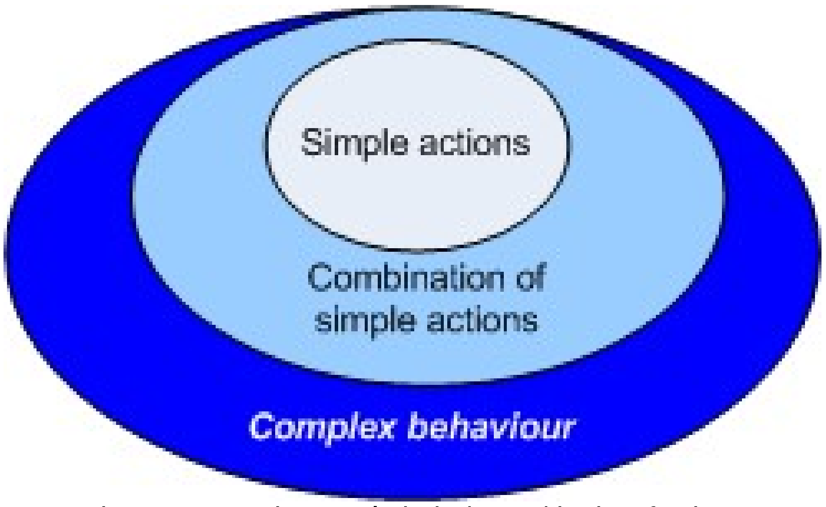


Figure 34: Increasing complexity in the combination of actions

4.4 Reasoning Functionalities

These functionalities comprise what can be called AIBO's brain. They goal is to reason with all the information that the system has available to inference a situation where AIBO can performance appropriate actions.

Since it was necessary to avoid the combinational explosion to make a flexible system, the nPME model as described in second chapter is not going to be used exactly in this system. Basically the components called *standards, preferences* and *goals* have been removed, and the main force that will drive robot's behaviour is based on **needs**.

When the system catches an incoming event, it will be translated for the reasoning part. Then, this event is going to put in combination with the different level of the needs to inference which is the need that has to be satisfied first.

Once the need is chosen, the next step is to choose the situation where AIBO encounters. If the situation let AIBO satisfy the need, the mood value will be increased positively and it will show positive emotions. This happens at the same time than the reasoning part orders the system to perform a set of actions which goal is satisfy that need. After that, the internal state of the robot is updated (needs gets their new value as well as mood and emotions) and the process starts again.

In Figure 35 a basic diagram is shown with the different reasoning functionalities in action.

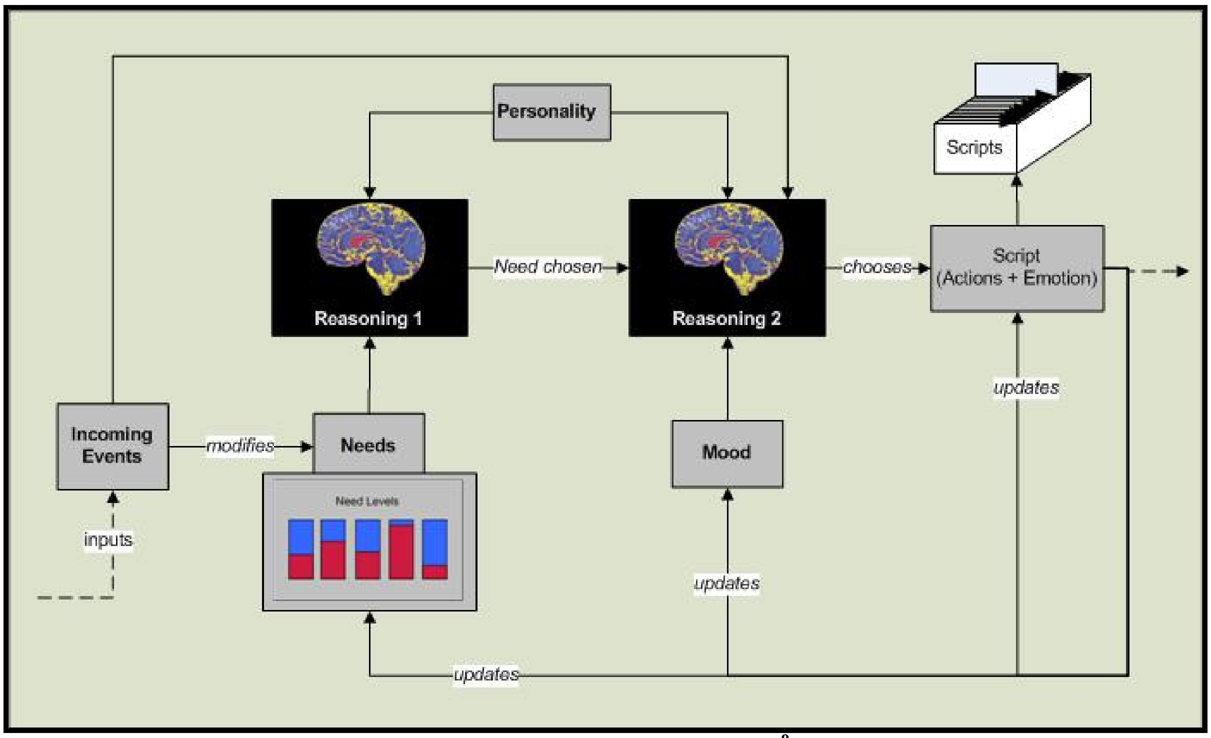


Figure 35: Reasoning functionalities. [9]

4.5 Complex behaviour: the real companion dog

As said above, the complex behaviour is reached when a combination of actions together with emotion reaction is shown. The best way to organize all the data, taking into account that there are thousands of different combinations, is to group basic actions with emotions in ***scripts*** (Figure 36), and a flow in time of different scripts will create the complex behaviour.

There is absolutely essential that these combinations will be coherent with the context where they take place. Since behaviours are driven by needs, the combination of scripts to get the complex behaviour will be done for each need.

[9] First inputs are introduced to the system as incoming events, that modifies the need values. Then a need to satisfy is chosen, and later according to the value of the mood and personality, a script will be also chosen. The execution of that script will make the system update internal values.

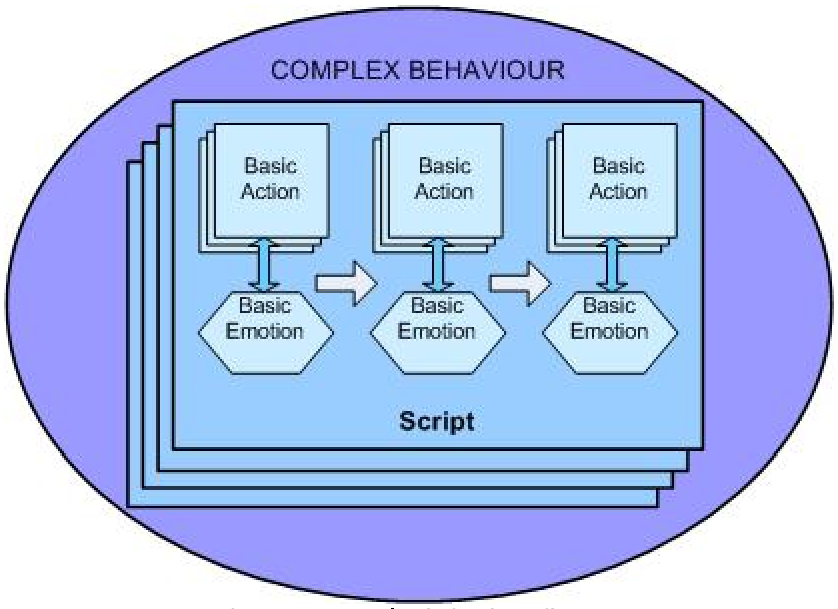


Figure 36: Complex behaviour diagram

4.6 System Design

Ideally the system is composed as it is shown in Figure 37. AIBO robot is used as a URBI server. A server is a system that provides some functionality when a client requests them. In our case, the client is a normal PC where the brain of the system will be running.

The robot theoretically will use its *Sensors* to catch events from external world. The event information is sent to the client. This will pre-process this information to translate it in a way that the rest of the system can work with it. In our system, as explained before, the events will be created by a Graphical Interface in the PC client instead of using AIBO sensors.

The *Reasoning Module* will use the information both from events and the emotional state (that is stored in the *Memory*) to infer the *script* it has to execute. With this, the client will request AIBO robot to perform some actions together with emotions.

The information flow is controlled by Central Control Module which is likely the main part of our system. CCU allows us to change a module and make the rest of the system remain without modifications. For example, if we want to substitute the Graphical Interface that creates events with another module that makes sound and image recognition using AIBO sensors, that would be possible with almost no change in the rest of the system.

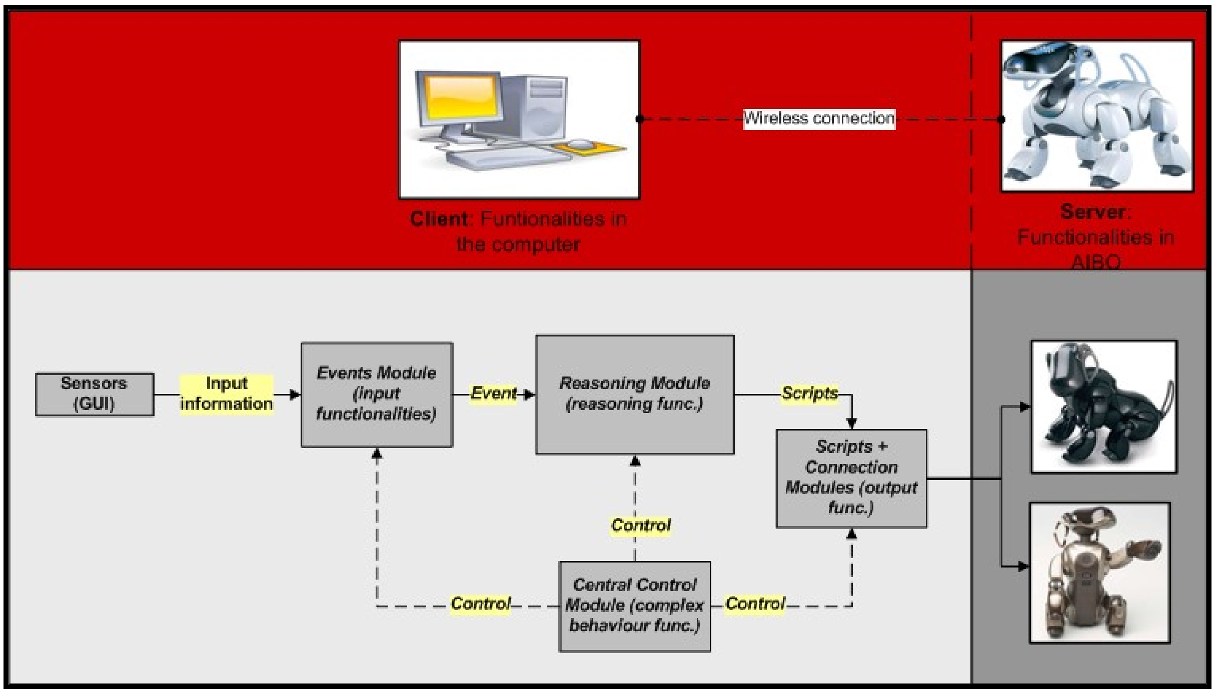

Wireless connection
Client: Funtionalities in the computer
Server: Functionalities in AIBO
Sensors (GUI)
Input information
Events Module (input functionalities)
Event
Reasoning Module (reasoning func.)
Scripts
Scripts + Connection Modules (output func.)
Control
Central Control Module (complex behaviour func.)
Control
Control

Figure 37: System functionalities overview

4.7 System Architecture

In Figure 38, a diagram with the conceptual design of the architecture of design is depicted. As we can see, the whole system is totally divided in different modules, and each one has a different functionality.

The idea behind this design is the fact that if someday is necessary to change one of the modules, this change will not very difficult. There is not direct communication between each pair of modules, except with the CCU. That is a big improvement from previous designs. All the information that a module has to exchange with another one, goes through the CCU, so one module does not need to know how the other module is implemented. The only thing the sender module has to know is how to send that information to the CCU.

Going step by step, let's explain each module more in detail.

4.7.1 Events Module

This module is the responsible of managing the events that are gathered by the system. From video and audio information to battery levels and timers will be controlled by this system.

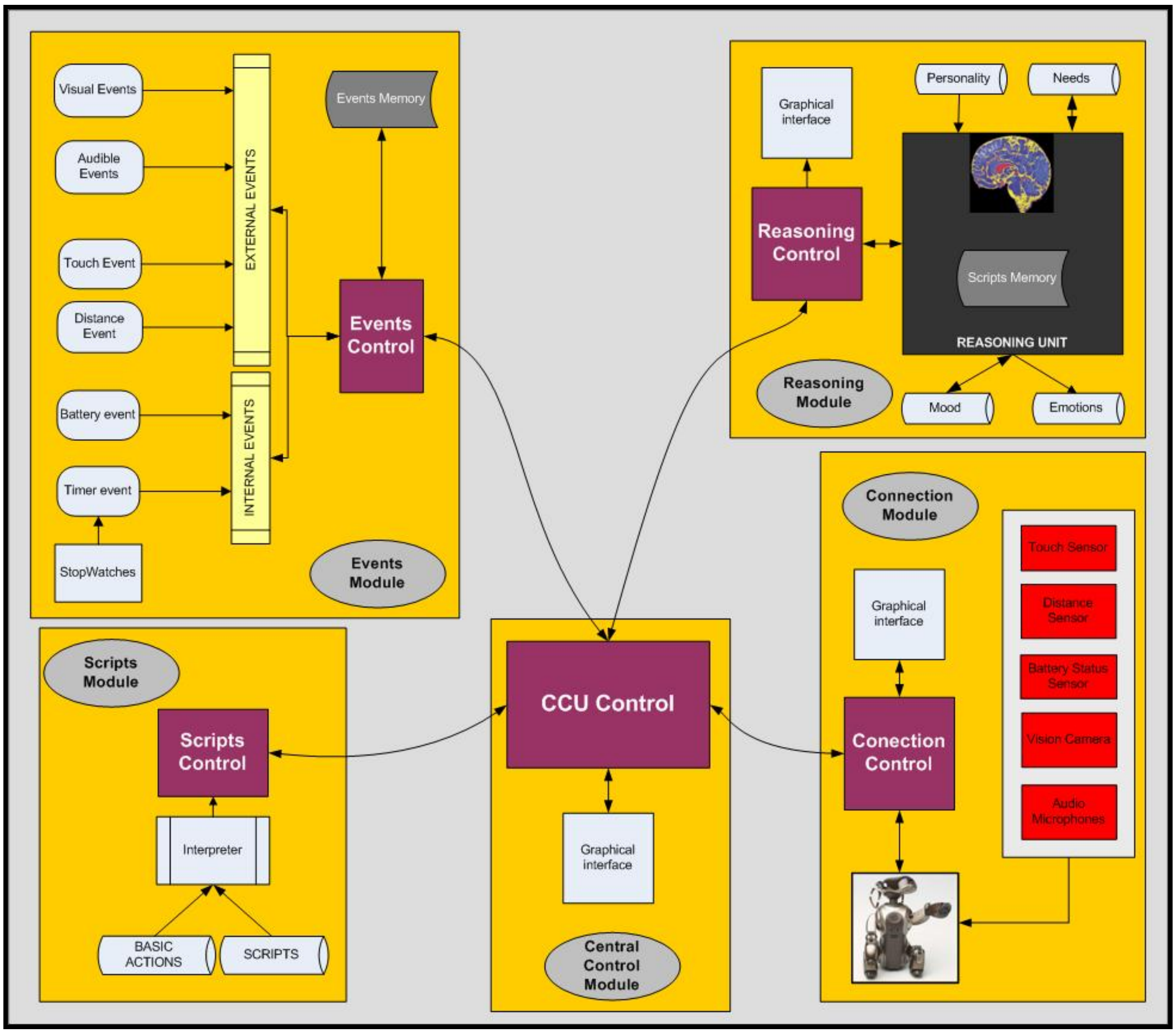


Figure 38: Whole System Architecture

The idea of this module is to take care of every event that occurs without considering the real source of that event. It does not really matter if the event was created by a GUI or by the robot sensors, because it will be processed in the same way.

The events that are going to be considered are:

- ***Video events***: images of people (owner, friends, etc).
- *Audio events*: commands, sounds (a door opening or another dog barking), characterization sounds to the robot (like 'good dog', etc).
- ***Touch events***: when the touch sensors of the robot are touched.
- ***Distance events***: these are used to know how far of some objects the robot is, but for our purposes, although the events are available, we are not going to use it.
- ***Battery events:*** to keep information about the levels of the battery. If this levels are so critic, the system will pass some information to the reasoning system through the CCU to act appropriately.
- ***Timer events:*** sometimes, the robot will have to wait for a response from its owner. In these cases a timer is needed. Depending on whether there is a response or not, the next sets of actions will be different. So some time limit must be available to evaluate if there was an answer or not.

The difference between external and internal events, is that the second ones are fired because of internal variables of the robot, while the first ones will occur when another actor or object performances something in the environment (like opening a door, giving a command to the robot or simply when an object is in front of the dog).

There is also a memory where all the active events are stored. It is important to mention that every event has an expiry time, that is, after certain time that event is removed from the system and will not participate in the reasoning process. That is obvious, for example in a situation where the robot hears another dog barking, but the sound finishes. If after some time, there is not image of the other dog, it probably means that it moved away, so our robot can ignore it and focus in other situation.

4.7.2 Central Control Module

That is the main part of the system. It controls everything that happens in every module.
On one hand, it acts as intermediate, passing information from one to another module. Sometimes it can make a little processing of that information before passing to the next module.

On the other hand, it also manages the main time line of the system, what means that it gives execution commands to every module at certain times. That is very useful in a real-time system because there are different modules that perform with their own time line (events can occur whenever, the robot lasts sometime in performing a set of actions, etc). With this central control is possible to avoid mistakes if, for example, one module requires some information that is still being processed by another module. Therefore is a very useful feature for maintenance of the system, especially if there is a module that has changed.

As seen above, the Central Control Module has to accomplish many tasks. It is composed by two main parts: a graphical interface and the core of the system, the Central Control Unit (CCU).

The *graphical interface* (GUI) is a simple display where it is possible init the system in two different modes: manual and automatic. The first one is just for test purposes. It initializes only the connection module where the user can introduce explicitly commands for AIBO server to make it perform simple actions. The automatic mode, also called **companion mode,** initializes the rest of the components of the system (events module, scripts module and reasoning module and of course the connection module). In this mode, the GUI also displays some messages into the screen giving information about the different process that are taking place in the whole system.

The *CCU* or *Central Control Unit* is the main control of the system. This centralized control lets the developer change one or more modules with very little changes in the rest of the system. This was made to achieve the needed flexibility.

The CCU (in companion mode) will be listening all the time to the incoming events. When a new one occurs it passes the information to the reasoning module. This will make the appropriate actions and will give a script back to the CCU.

In every moment the reasoning gives a script, the CCU translates this script into sets of URBI actions using the scripts module. When a script has been translated, the CCU sends that information to the connection module. This will send the right commands to AIBO server, to make it perform some actions.

Once AIBO has finished of performing those actions, the CCU recovers the control, and makes another iteration what means that the process continues again with the reasoning module. Events, as said above, are being listened during all the time. But thanks to the concurrent programming techniques both task can be accomplished at the same time. That is the main reason why our system is a real-time system, able to "see" and "act" at the same time.

4.7.3 Scripts Module

This module stores all the basic actions that can be performed by the robot and also the scripts that contain combination of those actions. There is an interpreter that will translate the scripts requested by the CCU into the robot's language to let the CCU send them to the connection module.

4.7.4 Reasoning Module

This is the most complex module in the system because it performs the reasoning with the *nPME* components as we explained above. Basically it has a control that will be connected to the CCU to make the reasoning at the right times and then it gives to the CCU the requested answer to execute some actions. As we said, this module will take personality components to inference a script to execute in combination with certain emotions, according to the receive events from the CCU. It has also a memory component to keep track of the scripts that have been already executed, avoiding incoherent repetitions.

Besides, this module has a Graphical Interface where information about the reasoning is displayed. Here, it is possible to see the level of the needs, as well as the chosen need to satisfy and of course the script that will be executed. This will help the user to check that the system is performing as desired.

In this section the personality parameters are explained. At then the reasoning process based in nPME model is explained in detail.

I. Personality parameters

These parameters were almost explained in chapter 2. However this explanation has some slightly differences with that one. So we include here the more precise description in relation with the personality model to implement.

i) Personality

Personality is based in five different traits (OCEAN model) that can take a value from 0 to 100 for each one. Since this is one of the causes of the combinational explosion, we have decided to define 4 different personalities (each one as a certain combination of those 5 traits). Depending on the personality, the system will chose one or another way of execution in certain moments. This means that one personality can show frustration when it cannot find its recharging base while another one can continue looking for it until batteries are off, for example.

Personality will determine by itself how to face different situations and how to perform one or another action showing different emotions, when necessary. These four personalities are: *friendly, aggressive, cowardly* and *disobedient*. The names have been chosen trying to be descriptive for a reader without psychological background. The idea behind it is to get an idea of their possible behaviour. Anyway, we provide an small description of them to clarify doubts:

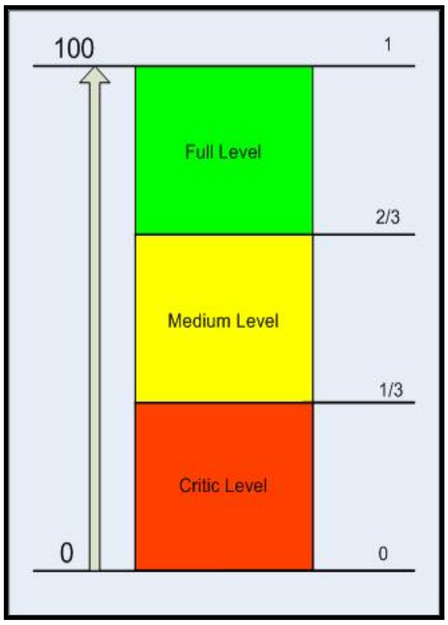
- ***Friendly***: curious dog, quite sociable and it is always willing to play. Normally it is friendly with unknown people/dogs/characters and non aggressive when things does not happen as expected.
- ***Unfriendly***: very shy with new items and it likes to play. It obeys to its owner, but it is quite aggressive with unknown people/dogs/characters. Although it is always willing to play, it gets angry easily when something does not occur as expected.
- ***Aggressive***: friendly with well-known people but very aggressive with unknown new items (people, other dog). It barks quite often and asks for food very angry. Normally it obeys its owner.
- ***Disobedient***: Curious with new items, and it likes playing. Normally it does not listen to its owner, and follows its own way. With the unknown people, depending on his mood, it will look or ignore them. When hungry, instead of feeling aggressive it will be sad.

ii) Needs

We will consider for our purposes only the first three needs from Maslow Model: *physiological, safety* and *love & belonging*. The satisfaction of the needs is based on Maslow Pyramid, so need 1 must be satisfied until certain level before satisfying need 2.

Each need is based in a *bucket* model. This means that each need has a value from 0 to 100. Through time that value is decreasing in the same way a bucket loses water with a hole in its basis. When the bucket has only a small amount of water, it means that the robot must satisfy its need executing some actions to fill the bucket again. It is some kind of homeostatic approach: every need must have certain levels of satisfaction.

For reasoning process the bucket is divided in three sections meaning three priorities when it must be filled:



- **Full level [2/3-1]**: it is not necessary to continue satisfying this need. Continue with higher needs.
- **Medium level [1/3 – 2/3)**: this need must be satisfied but only when the one that is currently being satisfied has finished. So basic ones will have higher priorities than superior ones in the same level, but only when the current one finishes its own satisfaction.
- **Critic level [0-1/3)**: Independently of which need is being satisfied, the need with critic level will immediately be satisfied (but only, of course, if it is more basic than the one currently satisfied).

Figure 39: Need bucket scheme

As a general idea, **needs** are treated as buckets that are going empty basically through time and must be filled when necessary. However some events can make a need decrease a lot its level. E.g.: when an unknown person appears at home the level of safety need will be critically increased, depending on the personality. This imitates **amygdale performance**. Brain amygdale is related with the feelings of fear and aggression and also it is related with some fast emotional responses. Decreasing the level of a basic need increases the probabilities of this need to be satisfied quickly, providing relative reactivity to the system.

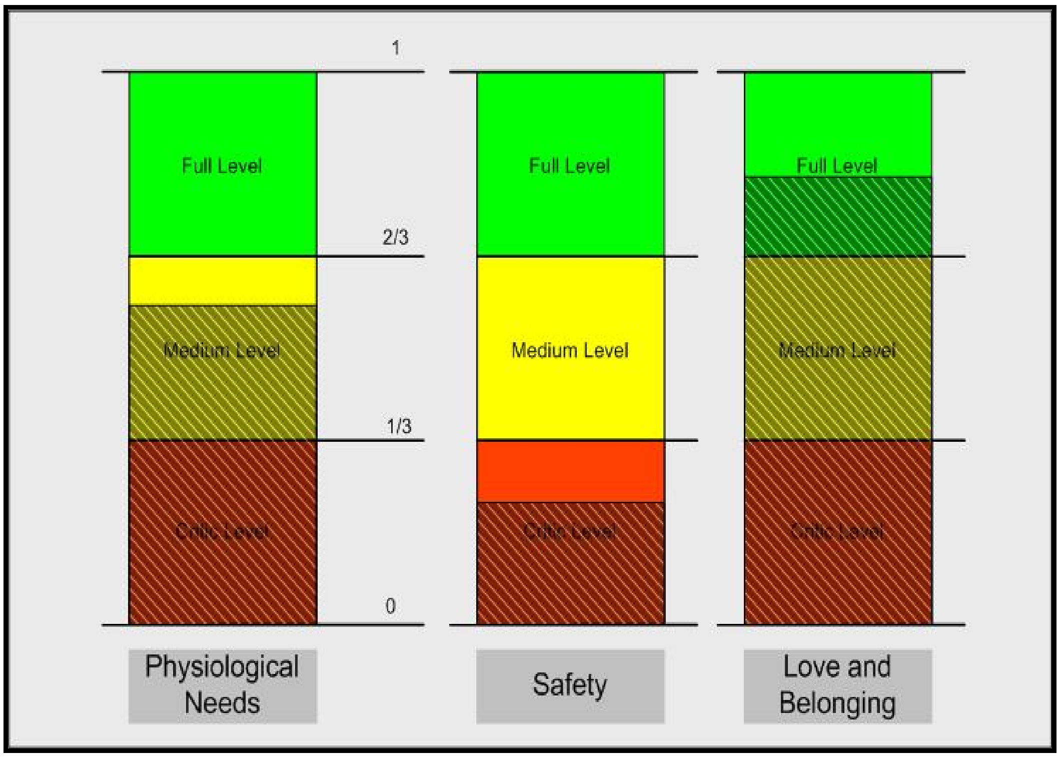


Figure 40: Needs priorities scheme

In this manner, every executed script will fill the respective need-bucket certain amount. That amount will depend directly on the 'meaning' of the script, taking into account if the execution of the script is something positive for the robot (it has found its recharging base) or negative (it has not found it).

iii) Mood

Mood is going to be use as a thermometer of the intensity of emotions. Mood will start with some certain value, and it will change continuously depending on the script that is executed. So if the script implies that the robot is satisfying one need, the mood will get better (higher positive value), and if the script is executed because it could not satisfy one need (*he wants to play and he does not find the ball)*, the mood will get worse.

The model of mood by Lang [26] with two axes (*valence* and *arousal*) has been simplified in only one axe meaning the intensity of the mood (arousal). Valence could be translated in the value of the arousal. If it is higher than 0, it means positive valence, and otherwise, negative valence.

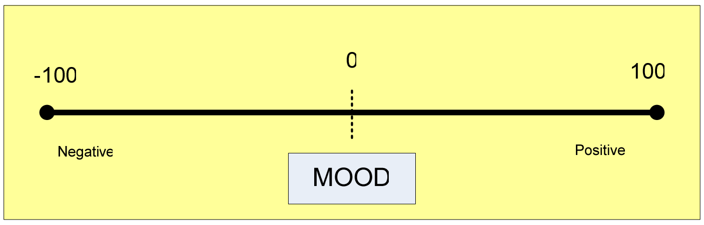


Figure 41: Mood scheme

iv) Emotions

The system will use the emotional states proposed by Eckman's model [21]. This means that the robot will show six different emotions: *joy, sadness, anger, fear, surprise* and *disgust.* In this way we keep what is used in [3].

Emotions will change always according to the actions that will be executed. Basically depending of the personality and the scripts of actions, the robot will show one emotion or another. Mood will provide the intensity of that emotion. So if mood is very positive and emotion is negative, it will result in a low negative value.

III. *Reasoning Process*

The reasoning process takes place in the *Reasoning Unit.* All the personality components are created and managed in the reasoning module, but the reasoning unit uses also events. These are coming from Events module through Central Control Module.

The reasoning unit takes all the components from nPME model and in combination with events gives the response to CCU. The reasoning unit has two main parts that are called inference engines. They make the reasoning itself.

The first inference engine first checks if there is any important incoming event. In that case it will modify the values of the needs to make them more precise with the current situation. Then the engine rechecks the values of the needs and it chooses the most priority one. That need is the one that must be satisfied immediately.

Now is the turn for the second inference engine. There is going to be a complete set of different rules for each predefined personality. That means that a certain personality can take totally different ways to satisfy a need than the rest of the personalities. So, now, with the appropriate set of rules, the need to satisfy, the value of the mood and the knowledge of the previous scripts (that were already executed), the second engine takes the decision about which is the next script to execute and which emotion will combine with it.

Once this reasoning has finished, the reasoning control sends the information with the script and the emotion to the CCU to execute in AIBO server. At the same time, the values of the needs, the mood, and emotions are updated according to the chosen script waiting for next iteration.

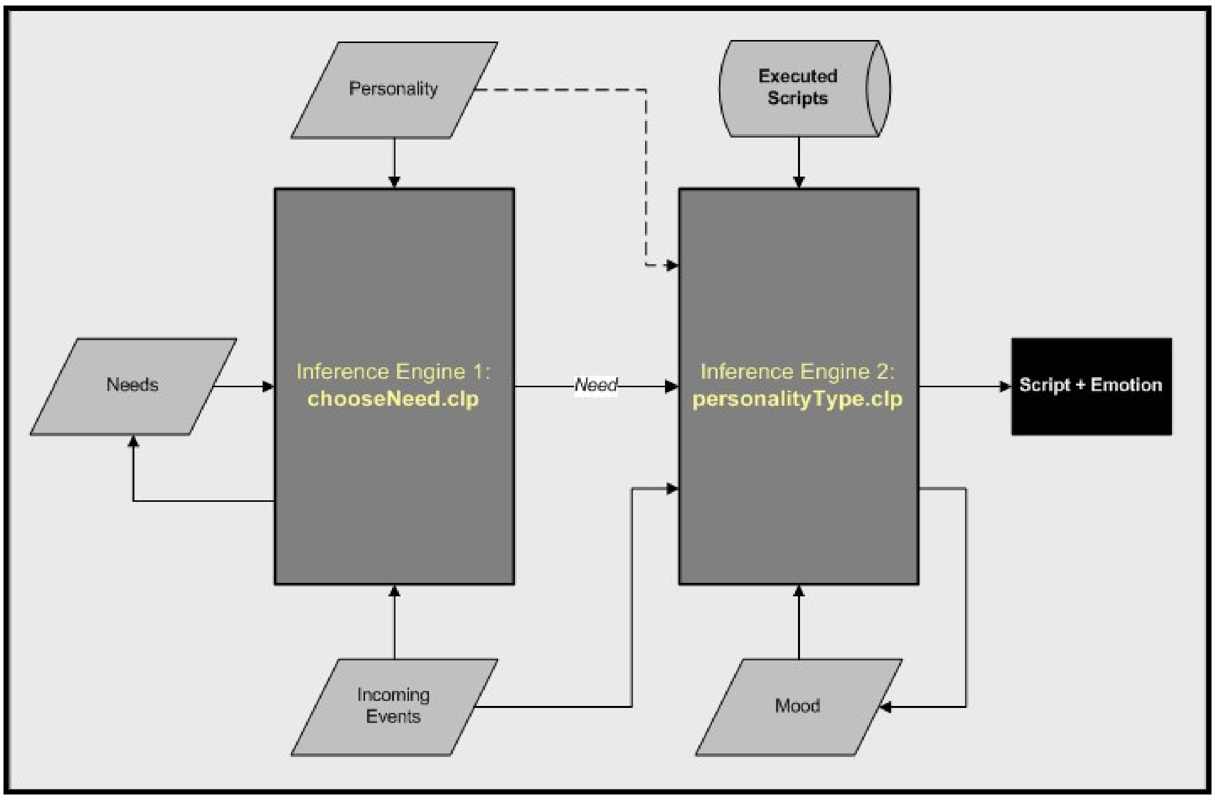

Personality
Executed Scripts
Needs
Inference Engine 1: chooseNeed.clp
Need
Inference Engine 2: personalityType.clp
Script + Emotion
Incoming Events
Mood

Figure 42: Reasoning Unit

All the information about the internal state of the reasoning unit is continuously monitored in a Graphical Display. It shows the values of each need in every moment as well as the values of the mood and the current emotion. It also shows the name of the script that is being executed. This gives enough information to the developer for adjusting the system appropriately.

4.7.5 Connection Module

It will manage all the connection with the robot. Every exchange of information between the system and the robot will be done through this module, in every direction.

Connection module will receive the commands (appropriately translated in the robot language) from CCU and it will send them to AIBO server. That will make the robot performance actions and show emotions.

Connection module has another functionality that due to time constraints will not be implemented. That is the capability of getting events from the exterior world through robot sensors and translated them into information that can be processed by the rest of the system. As we said in 4.2 external events will be wrapped with a Graphical User Interface. Every possible event can be created just by pressing a certain button.

However, this module has been designed with the premise that the substitution of the GUI with the real events caught by the sensors must be as simple as possible. It is not a trivial issue but the framework is consistent enough to let this inclusion.

5 Implementation

In this chapter the development of the software and the tools used for it are discussed. In the first section, some background about the different languages used is given with justifications about those choices. Second section gives an overview of the architecture of the system providing also a simple execution scheme over time that can be used as a guide for the full understanding of the system. The third section explains in detail every module of the system providing some UML diagrams to clarify how the system works.

Summarizing, in this chapter there is only the essential information to understand how the system works and the main reasons that drove us to take some decisions. More information about other tools, and more diagrams can be found in the appendixes.

Chapter overview:

5.1 Programming languages and tools

In this section there is a brief description about the programming languages and the software tools used to implement our system. The reasons why we took those decisions are also provided.

This project tries to develop a companion dog using a well-known robotic platform to test the coherence of the system. This robotic platform is the robot AIBO made by Sony. Since we need a robot that interacts with the real world, the project becomes the development of a real-time system, with the constraints that it brings. According to [39], a real-time system has to respond in certain time and in a predictable way to unpredictable external stimuli, and it has to fulfil some conditions:

- *Respect time constraints*: meet deadlines what means finish some tasks with time boundaries that it has to respect.
- *Simultaneous processing*: more than only one event may occur simultaneously, and the system must be prepared to face this situation and meet its deadlines.
- *Predictability*: these systems must react to all possible events in a predictable way.

In our case, we can say that our system is a ***soft real-time system***, what means that some delay in the performance is not absolutely critical, but of course that will increase the cost of the system, in the way that the good performance will also decreased.

Because of this reasons we need to use some powerful reliable tools, and a robotic platform that allows us to test the system whenever a change is introduced to test its performance.

For our purposes we decided to use a Windows machine and URBI to communicate with AIBO robot. The core of the system is implemented in Java, flexible enough to work with the rest of the software components, while the brain, that is, the reasoning part has been done using Jess, a language to build expert systems.

In next sections we give some reasons for our elections. For further information refer to Appendix C Additional Software Information.

5.1.1 URBI - Universal Robotic Body Interface

It is a universal robotics platform based on a scripted language designed to work over a client/server architecture allowing the user to control a system with sensors and actuators. The main features of URBI are:

- URBI has been developed taking care of the simplicity, so after some minutes one can understand the basics of URBI and start working with it.

- URBI is **a low level** command language. Motors and sensors are directly read and set. Although complex high level commands and functions can be written with URBI, the raw kernel of the system is low level by essence.
- URBI includes powerful **time oriented** control mechanisms to chain commands, serialize them or build complex motor trajectories.
- URBI is designed to be **independent** from both the robot and the client system. It relies on TCP/IP or Inter-Process Communication if the client and the server are both running onboard.
- You can control your robot with a URBI server running on it **via wireless LAN** from a normal Windows PC.
- URBI robot-specific server code and libraries are released under the GNU General Public License. The URBI Language, the URBI Kernel and the URBI Language Specification are protected by a separate and specific license.[13, 40]

The main component of the URBI framework is the **URBI Server** that is a C object compiled with gcc and copied on the memory stick. The server comprises of a **server kernel** that is robot independent and a robot specific add-on. The URBI server receives **commands** from a client and returns **messages** to this client. The normal way of using a URBI controlled robot is to send commands using **TCP/IP** on the URBI port (54000) and wait for messages in return. A simple telnet client is enough to do that for simple applications, otherwise libraries (**liburbi**) are available in most programming languages to wrap the TCP/IP sending/receiving tasks in simple functions. There is currently a **C++** and **java** version of liburbi if you want to control your robot using **C++ or Java** from an independent device like a PC/PDA and a **liburbi-OPENR** version if you want to recompile a **liburbi-C++** based program to let it run on the robot. [13]

A schematic approach to the way the entire URBI framework is composed is presented in Figure 43. The scheme is a representation of all alternatives in URBI rather then a concrete functional scheme of the system. [3]

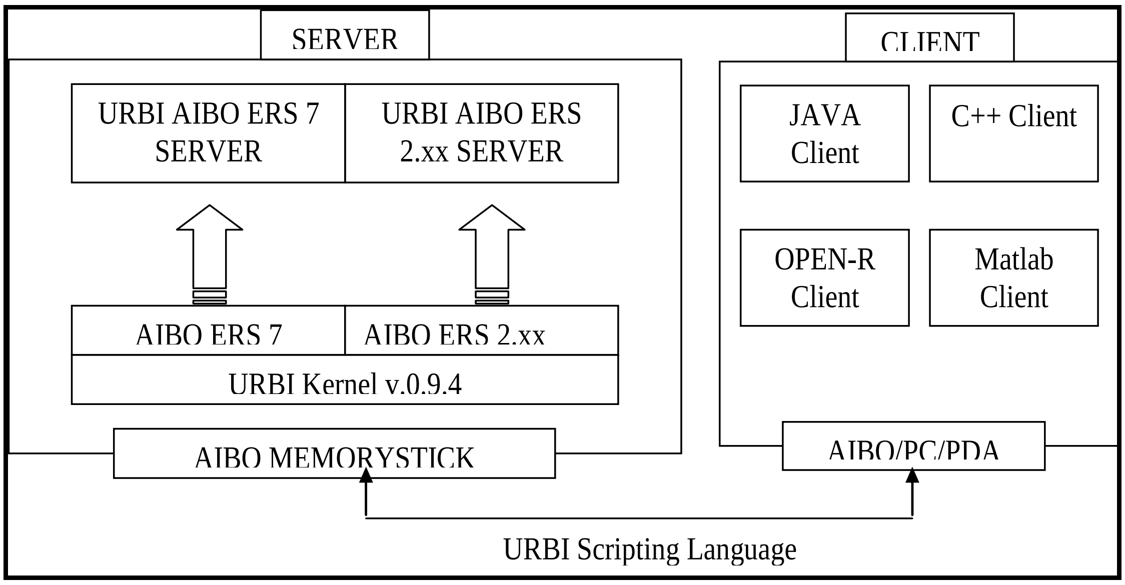


Figure 43: A schematic approach to URBI

The **URBI language** is the core of the framework and it defines a standard protocol to give commands and receive messages from the machine to be controlled. The code written in the URBI language that can be executed by AIBO are called **URBI Scripts**. [3]

Why use URBI?

There's no doubt that all development choices have their advantages and disadvantages. However one of the criteria became more important than the rest since Sony stopped AIBO production: reusability on other platforms and robots. URBI is an universal platform that by now can be used for AIBO, Mindstorms, Surveyor, irobot-create, ePuck, etc. Another features from URBi to make us take that decision are its ease of use as a programming language, the flexibility and the development speed. Besides it is a new solution, under development and continuous improvement.

We wanted to concentrate the attention of this project on a new personality model and its implications rather then concentrating on developing movements for a specific kind of robot. In this approach URBI helped a lot: first it is not dependent on AIBO and thus it will allow for the current prototype (with minor adaptations) to be tested on other robots also, and second writing scripts that make AIBO do actions is quick, easy to test and approachable even by non-software developers. Another aspect proffered in URBI was the division of actions in scripts. Thus AIBO's actions can be divided in small sub-actions that can be implemented by very small URBI scripts. Taking advantage of the fact that URBI scripts are represented as Strings in java, we created a library of scripts that can be concatenated in order to create complex AIBO actions.

5.1.2 JAVA

In the development process of this system one major programming language was used: Java version 5. There are intrinsic and extrinsic reasons for choosing Java over other development environments or programming languages (refer to *Appendix C*). However there is a reason that made us reject other languages almost from the beginning: the chance of reusing code.
The previous work on which this thesis is based used also Java [3]. At the beginning we thought that we could use almost all the code, but the implementation seemed not to be suitable enough for our purposes, and it was difficult to develop due to the design. We had to start from scratch, but there were some code (some classes) that were still useful for us only by doing some modifications. That is the main reason (although it has so many other advantages) that made us use Java.

5.1.3 JESS

The reasoning component of AIBO's mind is developed using Jess [41]. Jess stands for Java Expert System Shell. In short, Jess is an expert system (rule-based system, refer to *Appendix C*) that works with facts, and rules that are automatically triggered when the conditions are met. Jess is actually the Java version of CLIPS [42], with some added functionality to cooperate better with other Java classes. The fact that Jess is written in Java made the

choice of using it easy. It should not give problem to embed the Jess component in the rest of the application, written in Java. [3]
Jess uses a very efficient version of the rules finding facts algorithm, known as the *Rete* algorithm. Briefly, the *Rete* algorithm eliminates the inefficiency in the simple pattern matcher by remembering past test results across iterations of the rule loop. Only new or deleted working memory elements are tested against the rules at each step. The *Rete* algorithm is implemented by building a network of nodes; each representing one or more tests on a rule LHS. [43]
In the current prototype AIBO's central cognitive unit comprises of two coupled rule-based systems where the first one generates the fact-base for the second one. In the same way the second rule-based system modifies facts in the first one and resumes its execution. The two rule-based systems run in parallel if necessary and control each other's execution. The two expert systems present here extensively use shadow facts (Java Beans) and comprise of very large rule-bases that are set according to the goals of any specific application. [3]

5.2 Architecture of the system

This section shows the final architecture of the system that has been developed promoting the isolation of the different components to let future work with our framework.

The system has been designed to keep all the control from a normal Windows PC. This is connected to AIBO via wireless LAN connection (Figure 37). Although it has the disadvantage that without the connection with the PC, AIBO is only a piece of hardware without life, also that connection provides us some advantages:

- Total control over AIBO behaviour. In every moment we can see in the computer which the parameters and the variables of the system are, so it is easier to find bugs and correct them.
- Ease of use the robot thanks to Graphical Interfaces.
- Ease of modification of the code that states AIBO's behaviour. It is necessary to remember that the project is focus on creating a companion dog with coherence in its behaviour, and it is not only a programming exercise with AIBO, what means that continuous changes must be done during development process.
- This will let other researches modify the code in JAVA or in JESS to adapt AIBO for other purposes without the need of AIBO programming skills.

Using URBI helps us to keep that connection. URBI server is installed in AIBO, so it makes all the actions that the URBI client requests. This client is installed in the Windows PC, using the library *liburbi* for Java to keep the connection available.

We have added here the Figure 44 to give the first overview of the architecture of the system. This diagram will be explained in detail along the rest of the chapter, but for clarify purposes we can depicted here what are the main parts of the system. The CCU or Central Control Unit manages the whole system that is conceptually divided in 5 modules as explained in Figure 38 according to: inputs or events (*events_control, event_generator, events_definition*), the reasoning (N*eedsControl, ReasoningControl, PME_Components, Inference_Engines*), the output (*RobotOutput, ConnectionControl*), scripts (*ScriptsControl, ScriptsStoreRoom*) and of course the main control (*ccu_mainControl*).

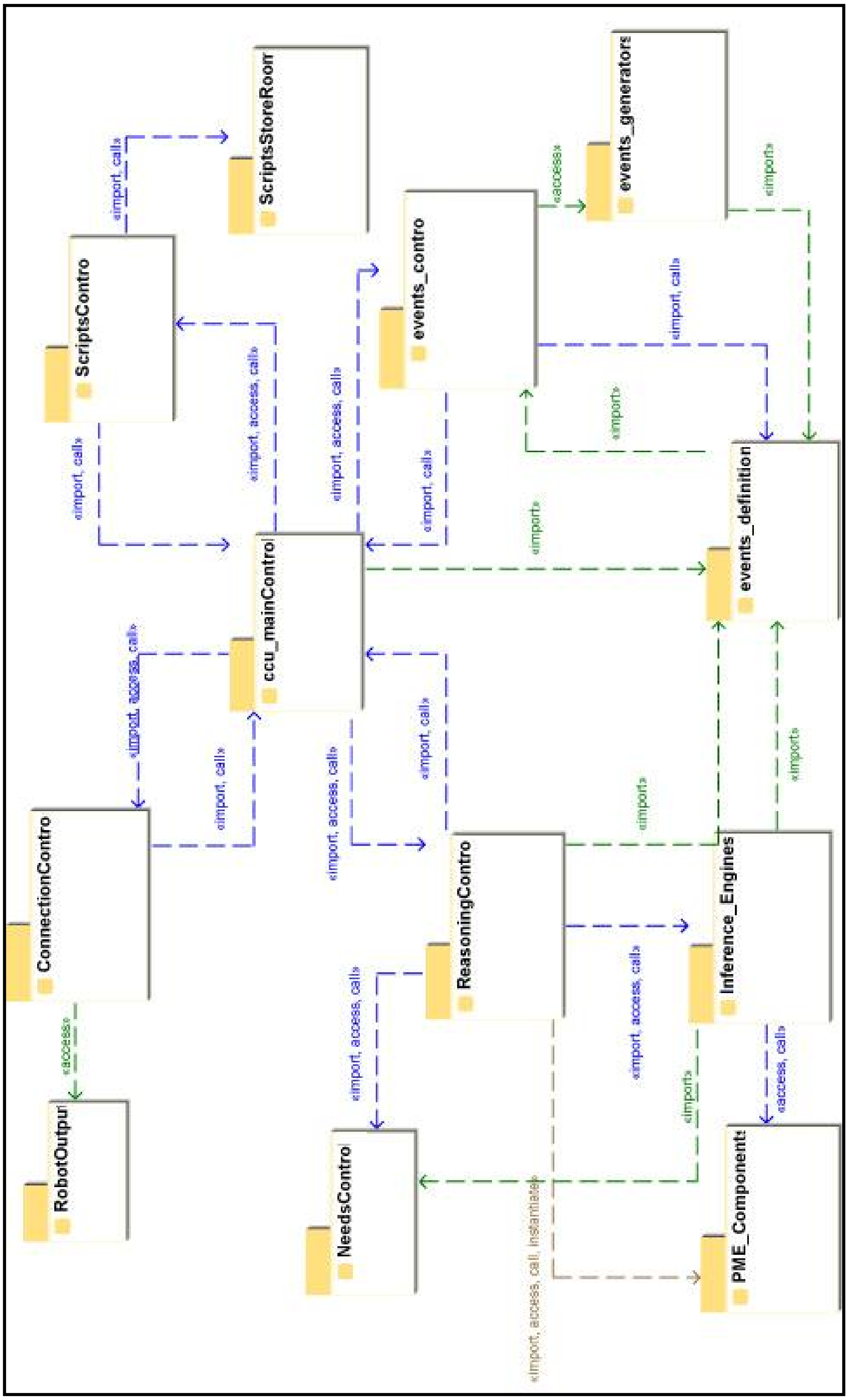


Figure 44: System Architecture

5.3 Execution Flow

In Figure 45 there is a complete diagram of the actions that the system must perform in order to understand the context of a situation and react appropriately. The diagram has been added for clarity purposes.

First, the whole system is initialized. This means that the main thread init another two more. These are the *mainGUI*, that manages the whole system and the *connectionControl* that controls the conection via wireless with the robotic platform. Thanks to the *mainGUI* the user can connect to AIBO via a graphical user interface. Here the user can choose between sending individual commands to AIBO or starting the *Companion Mode* that starts the automatic behaviour for AIBO according to the personality model.

When *Companion Mode* is enabled the system starts the *Reasoning Module* that has the goal of choosing certain scripts in determined situations according to incoming events and internal state of the AIBO (nPME model). It first connects to AIBO, and then starts an infinite loop managed in the *Central Control Unit* where all the automatic process is done.

Inside the infinite loop, the inference engines of the reasoning start working. There are two that work as threads: *chooseNeed* and *chooseScript*. The first one check if there is any event that changes the needs value. Then it chooses the most priority need and according to the fixed personality, it chooses one or another second inference engine. This one is the responsible of evaluate every incoming event and relates it to the chosen need. The result is the election of a script to execute in combination with one defined emotion. Every need chosen and every script executed is stored because they are important data for the next iteration of the reasoning. The reasoning control has another thread running when it is initialized: *reasoningGUI*, a simple graphical user interface that monitories the values of the parameters that influence the reasoning. Three more threads are also created in the initialization of the reasoning that control the timing of the needs, that is, they reduced their values each certain time slice.

When the script has been chosen, the *Central Control Unit* gets the code of that script. Using the functionalities of the *Script Control*, the script is translated into a set of actions, and each action into commands suitable for sending to the AIBO server. Those commands, one by one, are sent using *Connection Control* and more specifically, using *AIBOclient*. If everything is right, AIBO robot will perform some actions and emotions.
When the script has been sent totally, the CCU recovers the control and makes another iteration again, repeating the loop.

Whenever the user wants, she can create one event using the graphical interface embedded in the Connection Module. This has its own thread: connectionGUI. She only has to press one of the buttons of the events, and this event is automatically managed by Events Module, introducing it in memory and in the Reasoning Module. After certain time, the time of influence of one event, that event is removed completely from the system. That is controlled by a thread created for each new event.

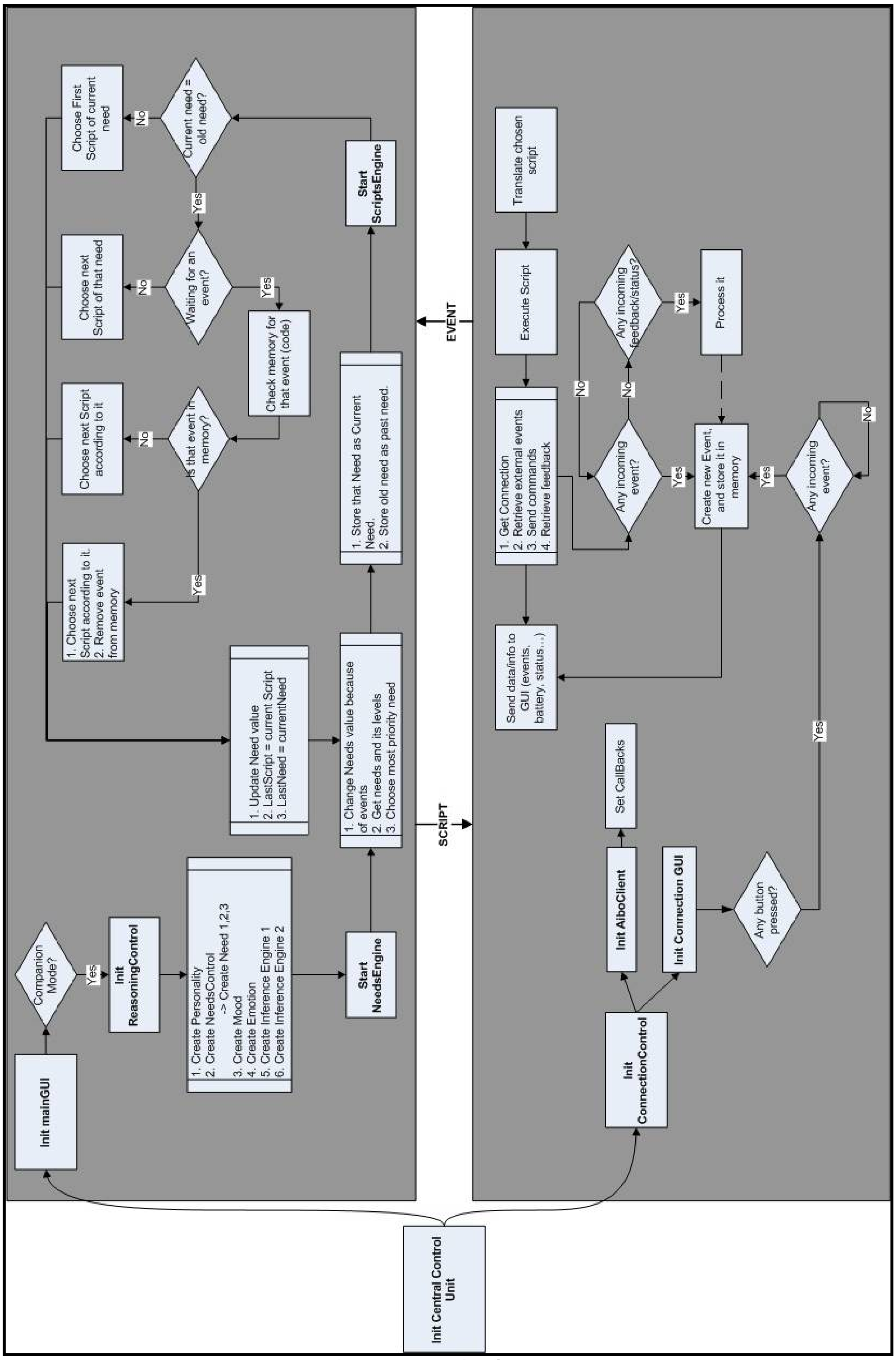

Init Central Control Unit
Init mainGUI
Companion Mode?
Yes
Init ReasoningControl
1. Create Personality
2. Create NeedsControl
-> Create Need 1,2,3
3. Create Mood
4. Create Emotion
5. Create Inference Engine 1
6. Create Inference Engine 2
Start NeedsEngine
1. Change Needs value because of events
2. Get needs and its levels
3. Choose most priority need
1. Update Need value
2. LastScript = current Script
3. LastNeed = currentNeed
1. Store that Need as Current Need.
2. Store old need as past need.
Start ScriptsEngine
Current need = old need?
No
Choose First Script of current need
Yes
Waiting for an event?
No
Choose next Script of that need
Yes
Check memory for that event (code)
Is that event in memory?
No
Choose next Script according to it
Yes
1. Choose next Script according to it.
2. Remove event from memory
EVENT
SCRIPT
Translate chosen script
Execute Script
1. Get Connection
2. Retrieve external events
3. Send commands
4. Retrieve feedback
Send data/info to GUI (events, battery, status...)
Any incoming event?
No
Any incoming feedback/status?
Yes
Process it
No
Yes
Create new Event, and store it in memory
Yes
Any incoming event?
No
Init ConnectionControl
Init AiboClient
Set CallBacks
Init Connection GUI
Any button pressed?
Yes

Figure 45: Execution flow

5.4 Modules of the system

In this section all the modules of the system are explained carefully from an implementation point of view. The relations between them will be shown as well as the information that exchange with the rest of the modules. Some UML diagrams are provided in this section, but we only include the diagrams that provide some functional information for the reader. More UML diagrams (especially sequence diagrams) can be found in the Appendix F.

The system is composed basically by five different modules. Each module is a set of different software packages. The main structure is depicted in Figure 46. Everything in the system is written using Java except some parts that were written using Jess and URBI. If it is not mention explicitly during each explanation, it means that part is written in Java.

The ***CCU module*** is the main module and manages all the system. Everything is controlled in first instance from this point. That makes it easier the possibility of doing some modifications in every module hiding details for the rest of the components.

The ***Events module*** will manage and store all the events that can occur. Those events are necessary for the reasoning part to keep track of the environment. That is the reason why that information will be available for the *reasoning module* via *CCU*. It is coded using Java but in certain way to let the system process this information in the Jess inference engines.

The ***Reasoning module*** is the responsible of creating AIBO personality what means the definition and modification of the different nPME components over time. It also reasons with that information together with incoming events to inference a set of actions (and emotions) that robot AIBO must perform. In this module we can find some files written using Jess, the expert systems language. The rest is also coded in Java.

The ***Scripts module*** is basically the store for the actions and combinations of those. Moreover it contains an interpreter to translate script text files into Java commands that are understood by the computer, and another one that translates them into URBI commands that are understood by AIBO server. This module contains plain text files, Java code and URBI commands.

The ***Connection Module*** is the responsible of creating, keeping and destroying the connection with AIBO robot via wireless LAN. For that purpose it uses a specified library provided with URBI called *liburbi* that lets Java control the connection perfectly. Every command that has to be sent to AIBO uses this module and all the information that AIBO server provides is processed first also by this module.

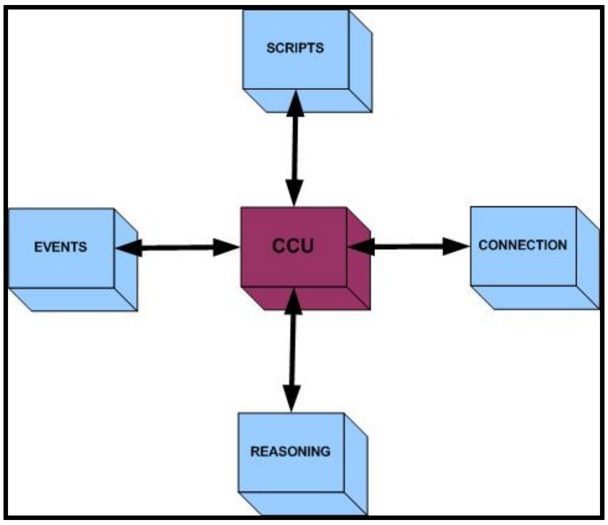


Figure 46: Modules overview

As said above Figure 46 presents the main modules of the system conceptually. From an implementation point of view, each module is composed by different packages that interact between them to give full functionalities to the system. In Figure 44 those packages and its relations are depicted in what is called Packages Diagram in UML notation. [37]

As we saw before, Figure 44 depicts a schematic view of the main packages involved in the system, showing common relationships from the point of view of the *CCU module*. Different modules contain different packages using the CCU module as an intermediate for exchanging of information. Table 14: Modules of the system with their packages shows the different packages that compose the system classified by modules. Specific information about each package, including more UML diagrams, can be found in next sections according to this classification.

Table 14: Modules of the system with their packages

MODULE	*PACKAGES*
CCU module	*ccu_mainControl* *AIBOExceptions*
Scripts module	*RobotOutput* *ScriptsControl* *ScritptsStoreRoom*
Connection module	*ConnectionControl*
Reasoning module	*Inference_Engines* *NeedsControl* *PME_Component* *ReasoningControl*
Events module	*events_control* *events_definition* *events_generators*

5.5 Central Control Unit (CCU) Module

This module is the central point from which all the system is managed. The main part of the program is located here and it controls the principal time line of the system. There are only two packages in this module: *ccu_mainControl* and *AIBOExceptions*.

5.5.1 Ccu_mainControl Package

This package only contains two classes CCU_Control and mainGUI. The first one is the one that control all the system (initialization, time-line of execution, exchange of information). The second one is just a simple graphical user interface where it is shown the evolution of the application.

I. *CCU_Control class*

As said above this is the core of the system. It is related somehow with the rest of the 'control' classes of the system, because that is its function: mainly it controls the rest of the system. Here you can find the *main* method that will starts all the rest of modules as well as the graphical interfaces.

It has been implemented as a Thread, allowing concurrent process while it is executing its actions. This is essential in a real time system, especially when at the same time, further than the robot itself, there are another three graphical user interfaces that are being executed. They have to refresh every certain time to let the user be informed about everything is happening and also let him interact with the system.

Basically this class performs next tasks:

i) Initialization of the system

When the system starts, this class is the responsible of initializing the other four modules of the system: *Scripts, Events, Connection* and also *Reasoning module*. This is necessary to make all the functionalities available for the precise moment. So this class will have direct contact, with all the 'Control' classes of the rest of the modules. This is done in this way to prevent any change of information that can be dangerous for the system. After this initialisation the CCU_Control starts the starts the **main Graphical User Interface** that monitors the system and let the user starts the *companion mode*, letting the system to complete the initialisation step.

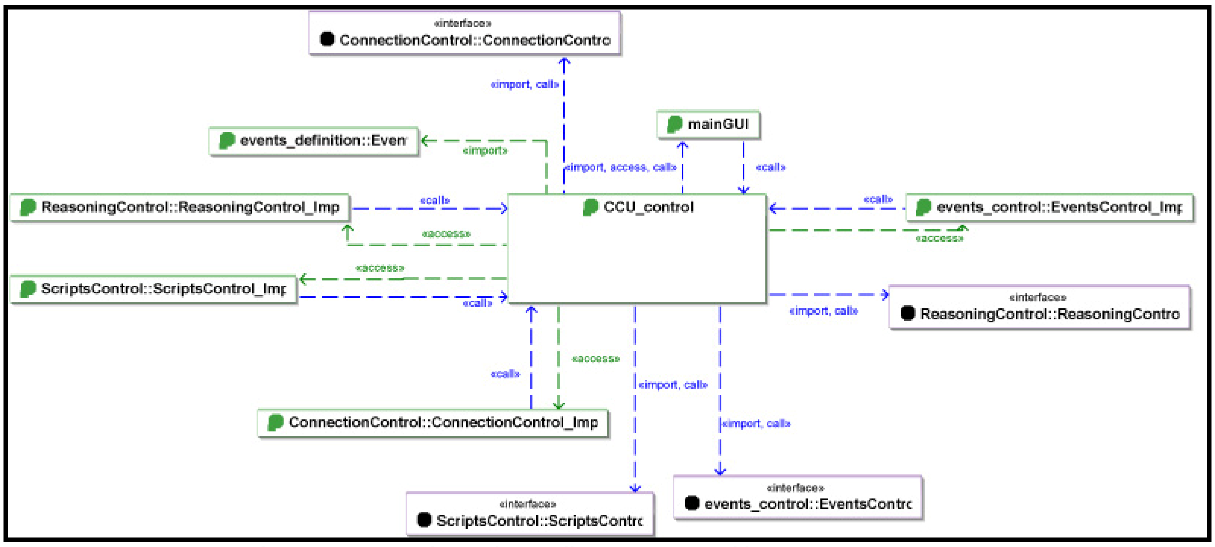

«interface»
ConnectionControl::ConnectionContro
«import, call»
events_definition::Even
«import»
mainGUI
«import, access, call»
«call»
ReasoningControl::ReasoningControl_Imp
«call»
«access»
CCU_control
«call»
events_control::EventsControl_Imp
«access»
«access»
ScriptsControl::ScriptsControl_Imp
«call»
«import, call»
«interface»
ReasoningControl::ReasoningContro
«call»
«access»
«import, call»
«import, call»
ConnectionControl::ConnectionControl_Imp
«interface»
events_control::EventsContrc
«interface»
ScriptsControl::ScriptsContrc
Figure 47: Class dependency diagram centered in CCU_control

Figure 47 shows the relationships of dependence that CCU_control maintains with the control classes of the system. As it can be seen, it only has relationship with the 'Control' classes, that is the classes that manages the inner performance of the rest of the modules. This is the key to keep a modularized design.

As it can be appreciate in Figure 47 every module is behind an *Interface* that is later implemented by another file with same name plus a tail like *_Impl*. The first idea for a software engineer is to build a common Interface for all the modules since they all send and received some information. However in our approach we a different interface for each module. We have done it in this way to improve reusability.[10]

It also important to mention that the class called *Events* is also used directly by *CCU_control*. The reason is that despite this belongs to another module, *Reasoning module* needs the information that this class provides (all the data of an event) through CCU.

ii) Control of the main time line of the system

The '**run**' method of the thread *CCU_control* is the one that contains the main functionality of this class, that is, it controls carefully the time line of the whole system: when the system needs some information, when to translate some script, when to perform reasoning, etc.

Probably the best way to understand how it works is to show a graphical representation of the execution flow, including the decisions, as it is done in Figure 48.

[10] If a new programmer wants to use part of our code, she has only take a view about which methods provide the Interface that really acts as an intermediary between CCU and the core of each module. So if someone wants to substitute a module, the only thing he has to do is keep the interface and make its implementation using its methods and data.

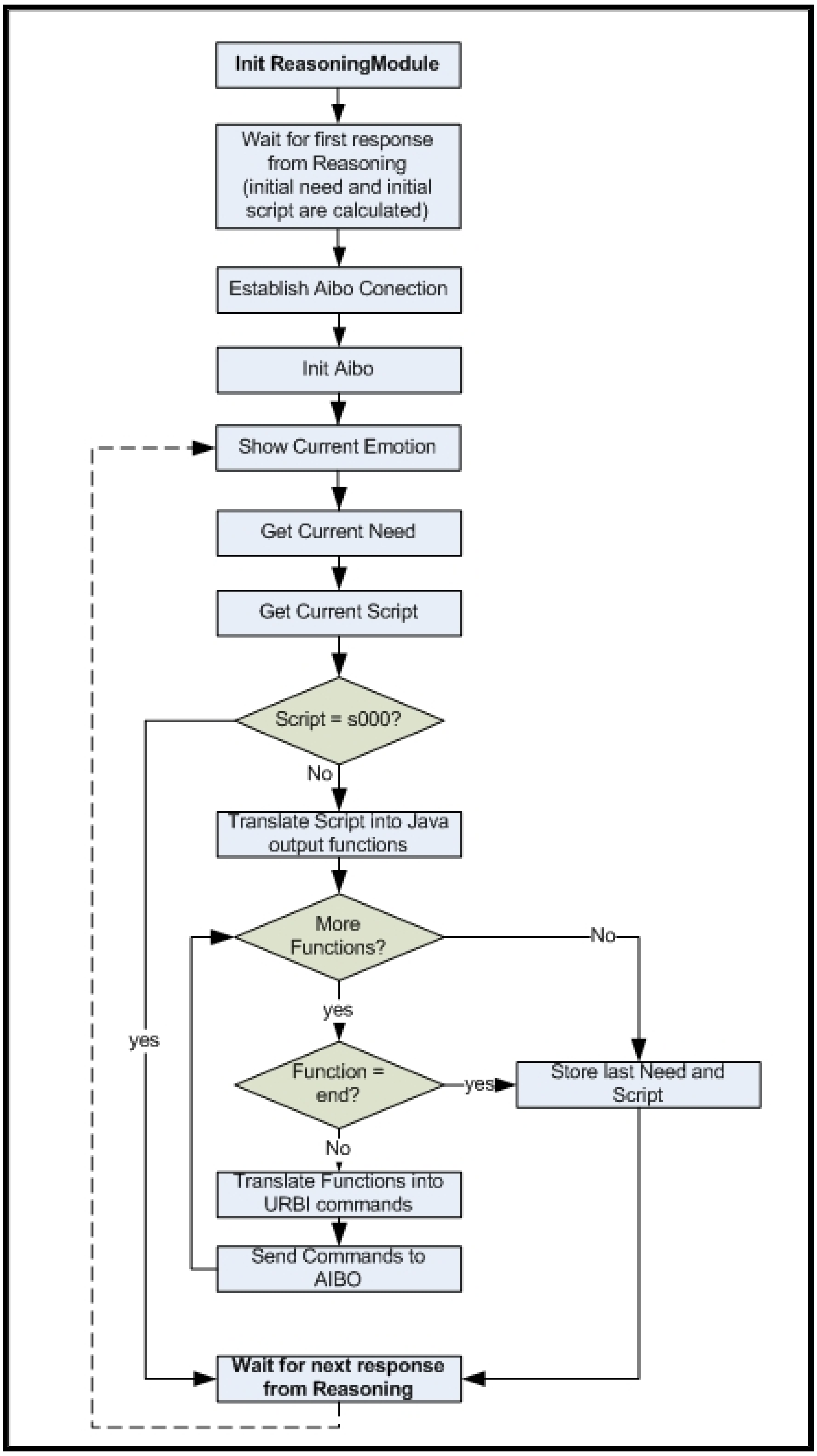

Init ReasoningModule
Wait for first response from Reasoning (initial need and initial script are calculated)
Establish Aibo Conection
Init Aibo
Show Current Emotion
Get Current Need
Get Current Script
Script = s000?
No
yes
Translate Script into Java output functions
More Functions?
No
yes
Function = end?
yes
Store last Need and Script
No
Translate Functions into URBI commands
Send Commands to AIBO
Wait for next response from Reasoning

Figure 48: CCU time line control

It is important to mention that some boxes in Figure 48 seem to be part of the reason module. Actually, we have created the CCU as a centralized class to control of the system. We wanted to have the main timeline totally managed from CCU. So from this class we make the initialization, and later in the main loop we make the calls to the Reasoning module (to get and send the current emotion, to get the current Need to satisfy and later the Script to execute) and the calls to the Scripts module to translate the Script into actions to send them to the Connection module.

More in detail: at the beginning the *Reasoning Module* is initialized, what means that needs start to decrease over time. Then the inference engines makes it first iteration for the reasoning, what means that the first need to satisfy is calculated together with the script that will try to satisfy that need. After this, the connection with AIBO via wireless LAN is established. AIBO is initialized and when this finishes, the CCU recovers the control and gets the type and intensity of the emotion. The emotion chosen is send to the server to make AIBO show it. This is done every time a new script is going to be executed. That means that there is a lecture of the emotion at the beginning of each script. Then, the CCU gets the data from the need and the script that the reasoning chose to satisfy. With the information of the script code, this script is translated first from a plain text file intro Java commands, and then from Java into URBI commands. When the whole function (set of sequential and simultaneous AIBO actions) is completely translated, that information is sent to AIBO server as URBI commands to let the server executes them. Once AIBO finishes the execution, the system makes another Reasoning iteration process giving next (maybe the same) need to satisfy and script to execute. During all the performance of the *companion mode*, the process of translation, execution and reasoning continues in an infinite loop. As it can be seen in the diagram, the chosen need and script are stored as input data in the inference engines for future reasoning.

iii) Responsible of the information exchange between the rest of the modules

CCU has another essential function to make the system robust and facilitate the task of debugging. It is the only intermediary between the different modules. This means that every exchange of information 'inter-module' is always done through the *CCU_control*.

As we have seen, all the modules have a control 'device', formed by an Interface and another class that implements this Interface. All these controls are managed by *CCU_control*. That means that this knows all the methods available in the Interfaces and it every control knows the public methods available in the CCU.

So when certain module request for some information stored in another module, this is requested to the CCU. The CCU has the appropriate method that will ask the other module for that data using only the methods that appears in the Interface. The control of the requested module will have another method that gets internally the information and it passes it though the control to the CCU. This will send the requested data to the first module.

For example in the case of the events, that is probably the most difficult one: when the connection module creates an event what it actually does is to call one method in the CCU that will call another method in the Events Control using the information provided by the Connection Control. When the new Event is created, the Events Control calls the CCU to put that event into the Reasoning Module. In this way, every new event will be used automatically in the next reasoning iteration. When the time of the influence of that event is over, Events Control gives the order to the

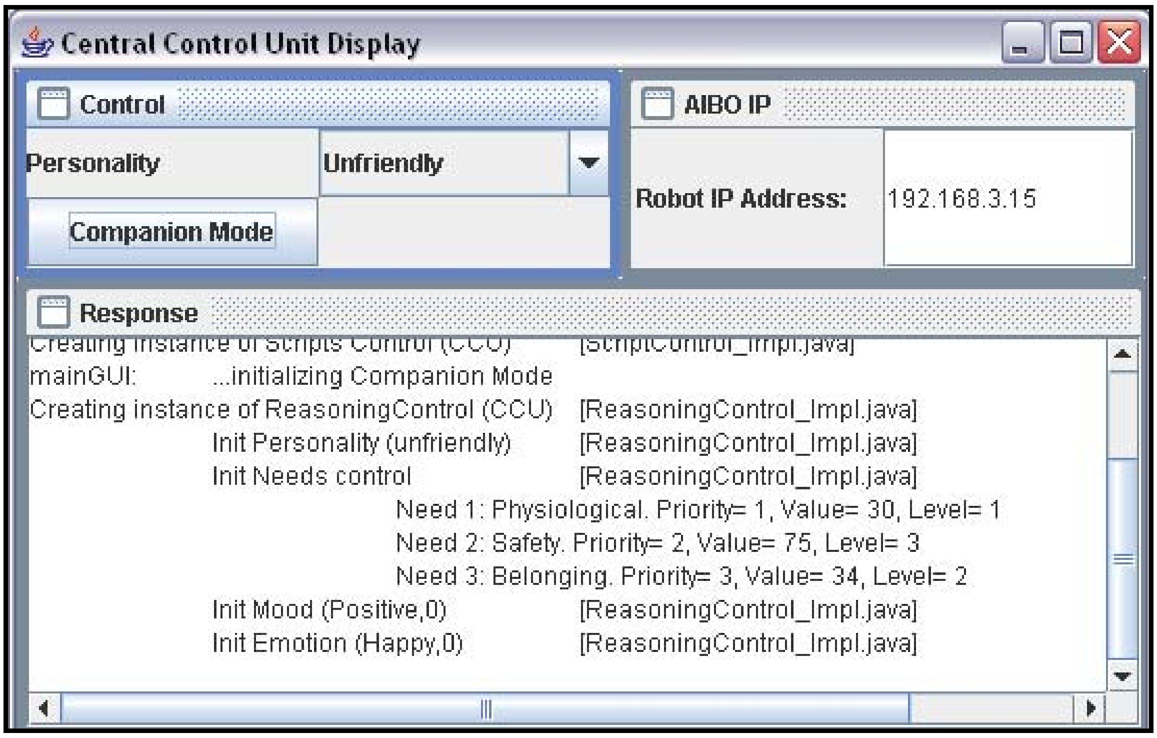
CCU and this gives the order to the Reasoning Control that removes that event from the working memory to avoid its use in next reasoning iteration.

II. *mainGUI class*

The mainGUI of the system is a very simple Graphical User Interface, as can be seen in Figure 49. This is enabled by *CCU_Control* during initialization part.

The functionalities of that GUI are:

1. **Start the Companion Mode** of the system, what means the automatic performance of the system following the timeline shown in Figure 48. It is done when the user presses the button called *Companion Mode*. This executes the method run of the thread *CCU_Control*. If the button is not pressed, the user can connect the AIBO using the *connectionGUI* in a normal manual mode as we will see later in Connection Module.
2. **Set the IP address** that the computer (AIBO client) uses to connect to the robot (AIBO server). This is used by *Connection Module*, however we include just in the case future developers do not want the connection GUI appears in the Companion Mode. An IP is always required.
3. **Show the performance of the system at any time** in a text window. The most important messages to let the user follow the actions than the whole system is executing are shown here. It shows the class that sends the message for debugging purposes.



Robot IP Address: 192.168.3.15

```
Creating instance of Scripts Control (CCU)       [ScriptControl_Impl.java]
mainGUI:          ...initializing Companion Mode
Creating instance of ReasoningControl (CCU)   [ReasoningControl_Impl.java]
                  Init Personality (unfriendly)       [ReasoningControl_Impl.java]
                  Init Needs control                  [ReasoningControl_Impl.java]
                              Need 1: Physiological. Priority= 1, Value= 30, Level= 1
                              Need 2: Safety. Priority= 2, Value= 75, Level= 3
                              Need 3: Belonging. Priority= 3, Value= 34, Level= 2
                  Init Mood (Positive,0)              [ReasoningControl_Impl.java]
                  Init Emotion (Happy,0)              [ReasoningControl_Impl.java]
```

Figure 49: Screenshot of mainGUI.java

5.5.2 AIBOException Package

This package only contains a set of classes with exactly same structure that defines many different exceptions for all the problems that can appear during execution.

One example of that is:

```
package AIBOExceptions;

public class EventNotFound extends Exception
{
    /**
     * Default Constructor for the exception
     * @param msg -message to be displayed in case exception is
thrown.
     */
    public EventNotFound(String msg)
    {
        super (msg);
    }
}
```

5.6 Events Module

Events module contains all the information about all the stimuli that AIBO can perceive in its environment. In Figure 50 the different packages of this module are shown.

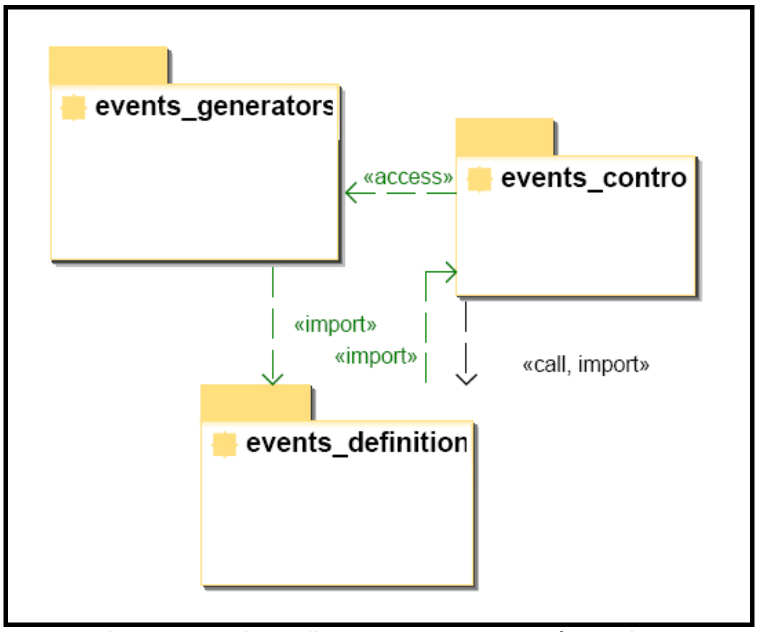


Figure 50: Package diagram – Events Control overview

Events Module is divided in three different packages:

- ***events_definition:*** where there is one class for the nature of each event that can occur. Logically they are organized according to the source (i.e. sensor of AIBO) that can detect it.
- ***event_generators:*** this is just an intermediate package to use the events_definitions from *events_control*.
- ***events_control:*** is the responsible of the communication and the managing between the Events Module and the rest of the system what is done through *ccu_mainControl* package.

5.6.1 Events_definition Package

This package only contains a set of classes to define the different type of events that can occur. In next sections we discuss a pair of details that are important for the well understanding of the program: first is how a numerical code of an event is built and second a special internal event called *Timer*. For more information refer to Appendix I.

I. *Events code*

The way this class uses to identify one or another event is a special code for each one. This code is part of the event definition, so every time the system needs to find certain event or execute some actions over it, it will use this code as a reference to find it.

Table 15: Code scheme for the events

SOURCE	SYSTEM	EVENT
2 digits	2 digits	3 digits

- **Source**: it is not an important factor for the design but is good for identification. It shows which is the source that creates (or detects) the events. There are only two in our system: *connectionGUI* -1- (it will be explained later) and AIBO sensors -2-.
- **System**: it means the nature of the event, or in other words, the sensor related to that type of information:

Table 16: Code for an event depending on the sensor

System	Code
Audio	**1**
Video	**2**
Touch	**3**
Distance	**4**
Acceleration	**5**
Battery	**6**
Internal Timer	**7**

- **Event:** it shows the name of a particular event given the type of information. *For example: a command from its owner meaning "come here".* (For more detailed information, refer to Appendix I.)

II. *Timer Events*

Sometimes, during a situation, AIBO has to wait for an incoming event. When this happens the last action of the set for that situation will fire a timer. So Timers will also condition some possible incoming situations. If a timer is launched, the CCU will check if new events match with one of the timers, and if that happens, the CCU will take the necessary measures.

The ***general inactivity timer*** starts counting when the dog finishes the last action of a situation. If there is one event to make AIBO react before the timer has expired, the timer will be removed and AIBO will continue its normal behaviour. If the timer expires before an event occurs, this will be the signal to choose one specific situation (there can be several randomly chosen situations) to show AIBO's *Boredom*.

Some different ***waiting timers***: there are situations where AIBO is waiting for a new specific event (*AIBO goes to its owner and expects a stroke. The stroke can come or not*). When it starts to wait for a predictable event the timer is thrown. If that specific event appears the timer is erased and AIBO choose one specific situation (*it moves its tail happily showing some combination on LEDs*). If there is no event, the timer will expire making AIBO choose one specific alternative situation (*it turns around and goes to its place angrily).* The actions that are chosen because of a timer will be conditioned both to the event itself and the previous situation.

Therefore timers add the possibility of anxiety in the dog behaviour, and make it more complex and realistic.

5.6.2 Events_control Package

This package is the responsible of the control of everything related with the events. It contains several classes that deserve to be explained separately. For a fast overview including its relations, we present a class diagram in Figure 51.

EventsControl is the class that controls totally the production, modification and elimination of events. When a new event is created, it calls an appropriate method in one of the classes of *event_generator* package. To select one or another method, the code of an event is used. So this information should be known by the Connection Module (or at least by the CCU).

Besides when a new event is created *EventsControl* creates a ***Stopwatch*** object and introduces it as part of the event. This is done because normally events have only some time of influence. So after that the event should not influence the behaviour of the dog. In that way, when an event is created, it is automatically introduced in a memory (***EventsMemory***) using its code, and the stopwatch starts running. Then, the event is included using the CCU into the Reasoning Module. When the stopwatch finishes, *EventsControl* removes the event from memory and it gives to the CCU the appropriate command to make the Reasoning Module remove the event also from working memory. So with this simple procedure, event is not longer user for the inference process.

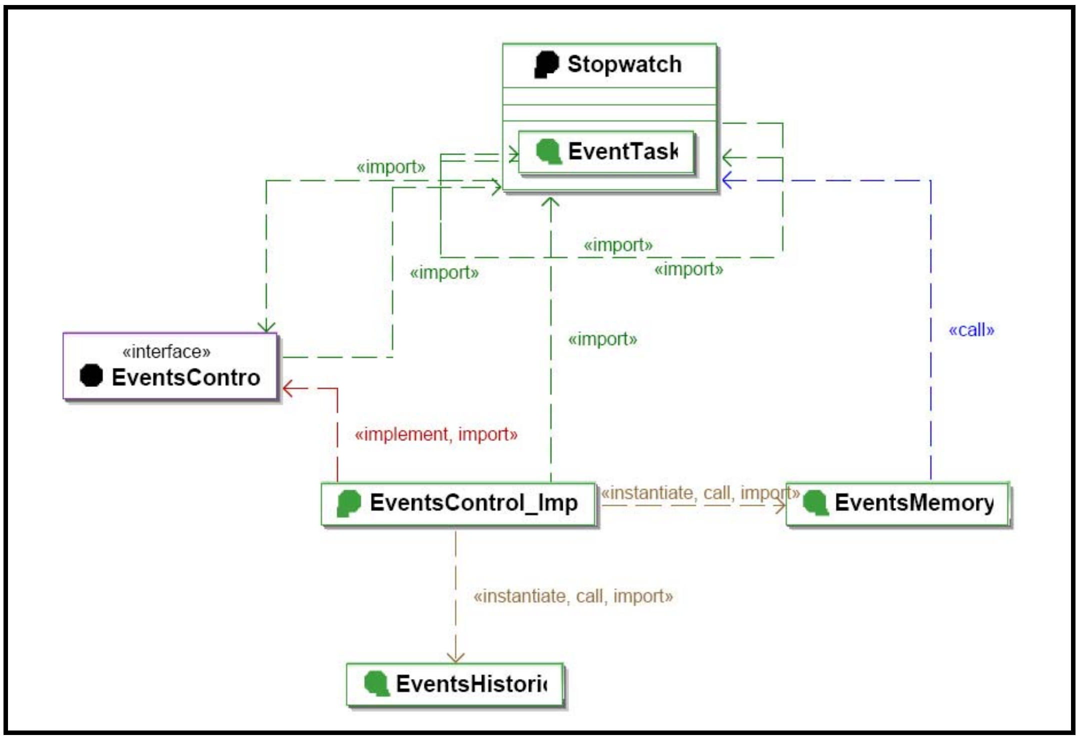


Figure 51: EventsControl class diagram dependencies

5.7 Reasoning Module

The Reasoning Module is certainly the most complex part in our system. It uses the power of the combination of two Expert Systems (using Jess) to implement the process that creates the personality of our companion dog.

This module, as the rest, has a direct relationship with the CCU Module, because if the CCU is the main component of the control, the Reasoning Module is with difference the implementation of the concept of this thesis. In that way, despite the fact we explain here the particularities of the implementation of this module, the logical combination of the components to choose the right behaviour is further explaining in the Design section of this thesis.

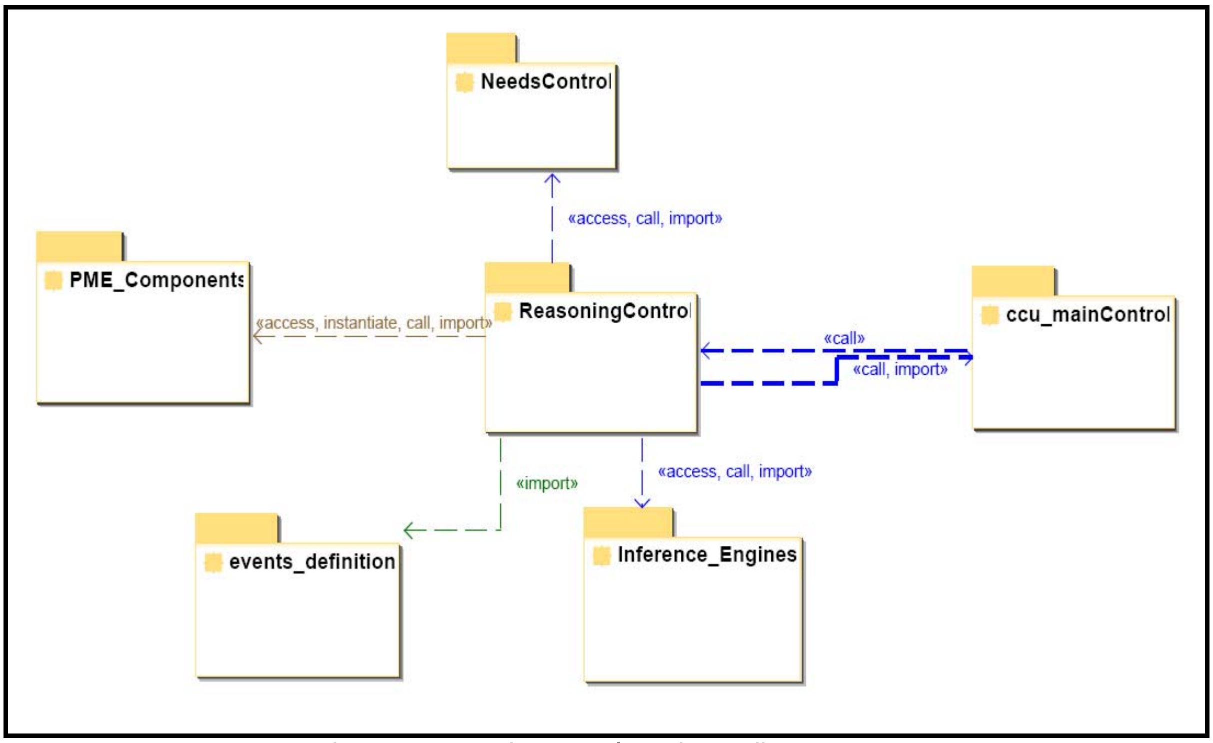


Figure 52: Reasoning control - packages diagram

As it is shown in Figure 52 this module is composed by different packages. *NeedsControl* and *PME_Components* are used as data for the inference process in the Expert Systems. The reasoning process itself is coded under the package *Inference_Engines*. The creation of the components as well as the control of the engines is done by *ReasoningControl* package. Let's explain those packages more in detail.

5.7.1 ReasoningControl Package

In Figure 53 it is depicted a complex class diagram centred in *ReasoningControl* Interface and its implementation. We start explaining this diagram in this section and continue in next ones. The first thing to say is that the relationship between the classes of the different packages is quite narrow what makes more difficult the task of implementing. However this gives more robustness to model of personality since there is not a trivial issue.

The ReasoningControl package has only three classes: ReasoningControl, ReasoningControl_Impl and reasoningGUI.

ReasoningControl is the responsible in the first instance of the creation of the different nPME components that will be used for the reasoning. So when this module is initialized by the CCU, those components are created and a new graphical interface is depicted in the screen to show information related to the inference process.
Besides, this class has many functions available for the CCU to recover data (current Need being satisfied, script that has to be executed) and to modify some values when needed.

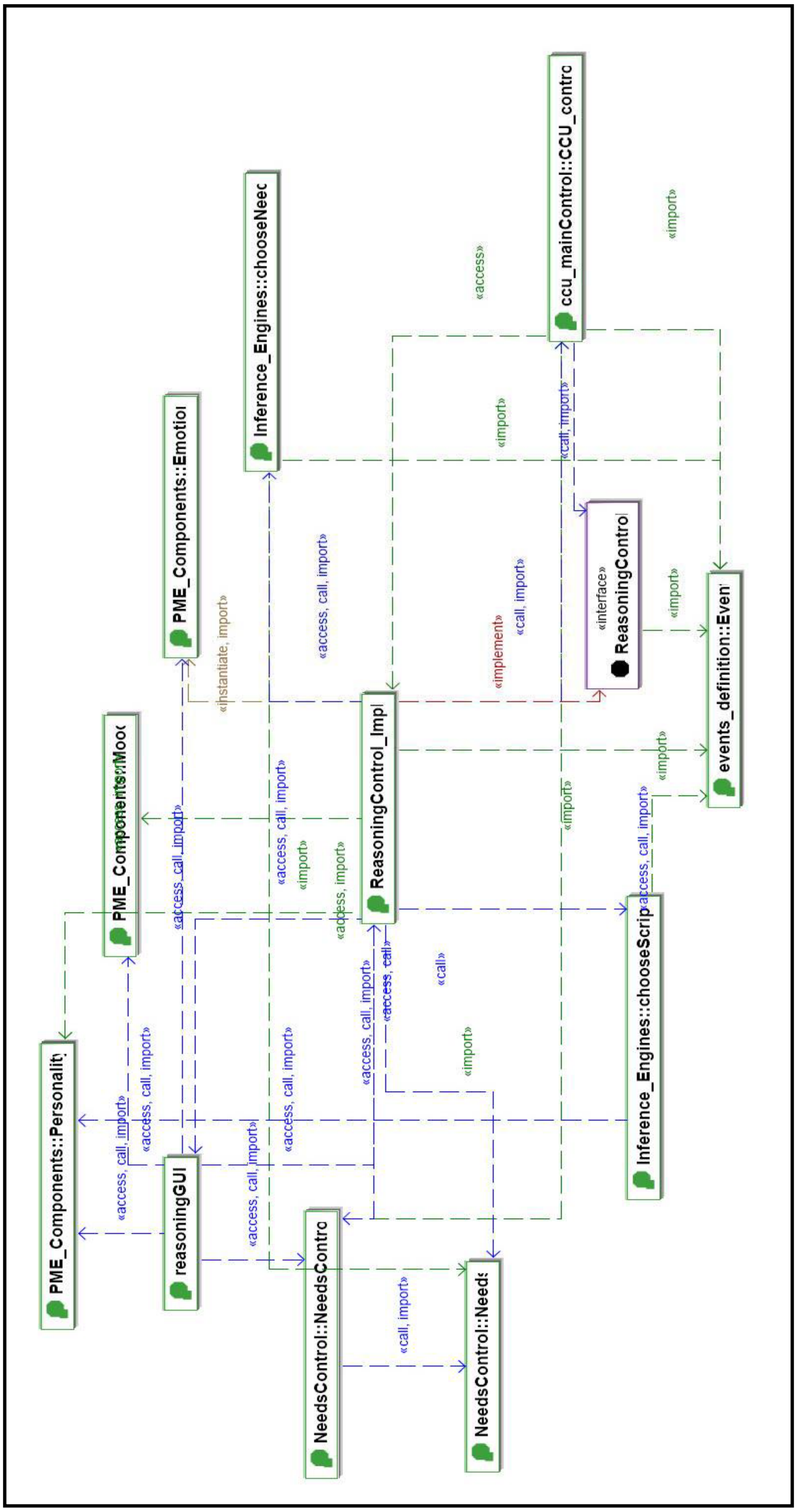
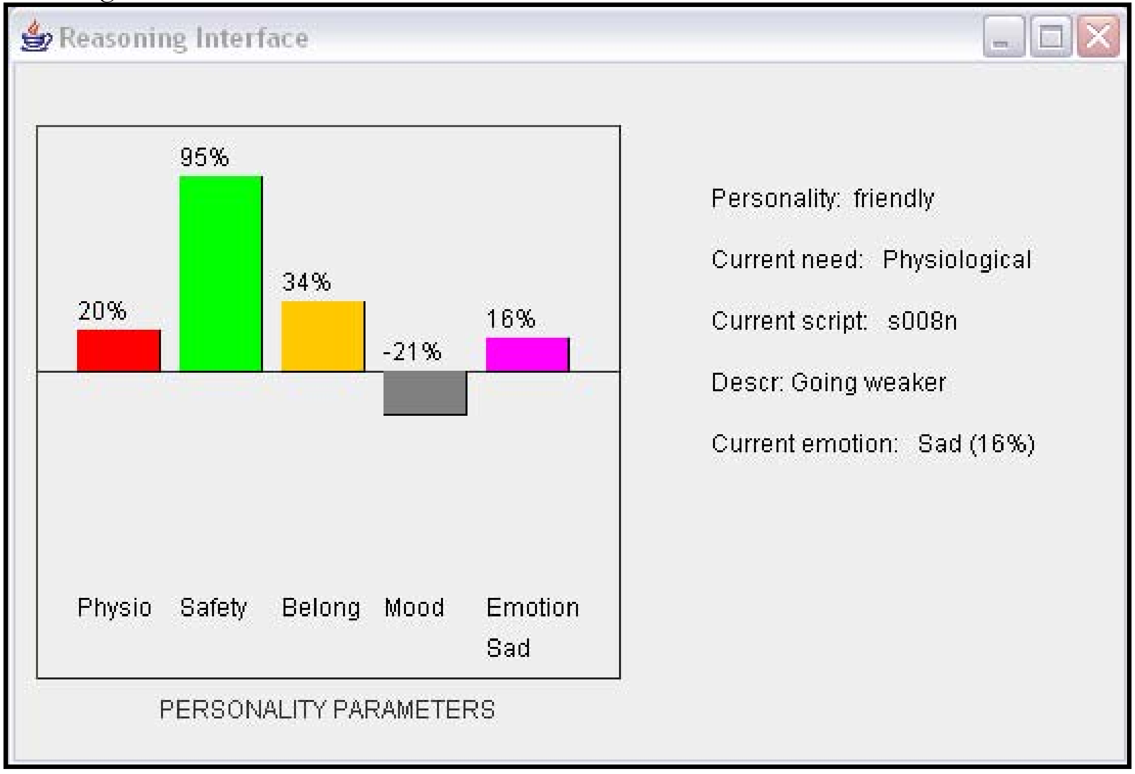


Figure 53: Reasoning Control class dependencies diagram

ReasoningGUI provides the user with some important information to follow the reasoning process, as can be seen in Figure 54. It shows the current level of the needs in different colour showing this level (critical, medium and full), as explained in the Design section. It also shows the mood in certain moments with its value next to the bar meaning the level of certain emotion. The personality chosen by the user is also shown together with the name of the need that is currently been satisfied and the code of the script that AIBO is executing.



Personality: friendly

Current need: Physiological

Current script: s008n

Descr: Going weaker

Current emotion: Sad (16%)

Figure 54: Screenshot of reasoningGUI

5.7.2 PMEComponents Package

This package comprises some classes designed to create the first components to define a personality: *personality, mood* and *emotion.*

The code of these classes is quite simple because it is only for definition with some methods to let the modification of certain values when the Inference process requires. They all uses Java Beans Patterns (as explained in the Appendix C) to let Jess use the components defined in Java for the reasoning process. So the three components follow that specific structure.

Personality uses a text file with its five traits to define which type of predefined personality it uses. By now the definition of one or other personality is very rude, but in the future another inference system can be included to create more variety of personalities.

5.7.3 NeedsControl Package

In the Design it was shown that each need is modelled as a bucket with a certain level of liquid. This level decreases during time or when some unforeseen event occurs. And the only way to fill each bucket is executing some scripts that satisfy that need in particular. This is the reason that the implementation of the needs is more complex than the rest of the PME parameters. The class diagram is shown in Figure 55.

They have their own control class *NeedsControl* that will take care of creating one stopwatch (*NeedsClock*) for each need to allow the 'empty' process each certain time. When the value of zero has been reached, the need keeps that value until one script satisfy it. *Needs Control* is the intermediary again between *Needs* and the *InferenceEngines* package.

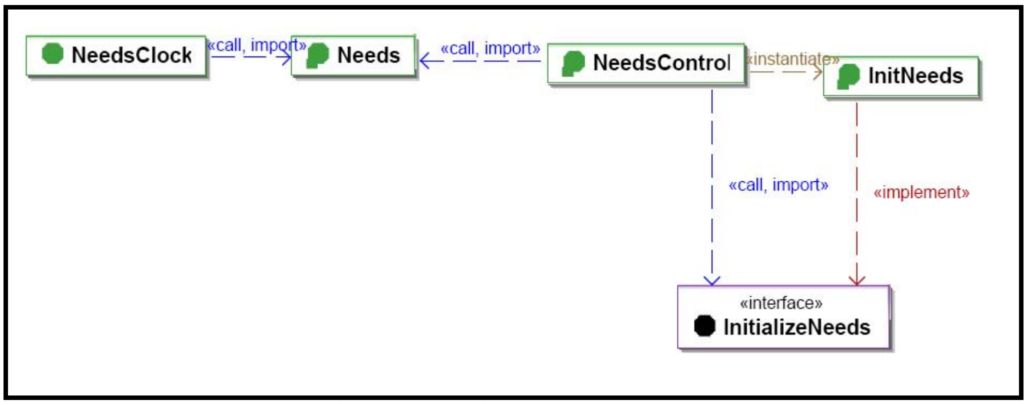


Figure 55: Needs control class diagram

It is important to mention that needs have to be also used in the inference process so they also follow the Java Beans structure defined for Jess.

To get the initial value for the needs, a plain text file is read using *InitNeeds* class.

5.7.4 InferenceEngines Package

This is the main part of the reasoning system. Its function is to reason with *Personality, Mood, Emotions, Needs* and *Events* to estimate the right script. That script wraps the actions for AIBO to execute including the emotions to show. By now two coupled inference engines will be used to do all the reasoning process for choosing one reliable script. It is in this package where the Expert Systems coded in Jess are defined and controlled by Java files.

To control the engines two different Java files have been created: *chooseNeeds* and *chooseScript.* Each one will control one inference engine, both the execution and the data that must be stored and recovered from one iteration to another.

If the reader checks the files of each package, it can be seen that there are more that two Jess files. The reason is that while the First Inference Engine has only one possible definition, the Second Inference Engine has by now until four different implementations, one for each fixed personality. This separation makes that the debugging of certain personality becomes quite easier.

As explained conceptually in the design, the first inference engine has three different parts:

1. **Events fast reaction**: the system checks if certain events are in the working memory. They are special events that affect one or more needs. So if some of them exist, several rules will be fired at the beginning having as result the change in the values of those needs. This is what we called *amygdale approach*.
2. **Need election**: The second part evaluates all the level of the needs and according to their values and the previous one, the most priority one will be chosen. A table with those elections can be seen in the Appendixes.
3. **Change to Second Inference Engine:** This is done simply by looking to the personality and only when the other two parts have finished completely.

The Second Inference Engine has three different parts: one per need (depending on the previous inference engine). Inside each one, the rules have always the same structure:

event + previous script + mood value
=>
next script + emotion (intensity) + change of need + change of mood + erase event

5.8 Scripts Module

This module has a simple well-known function. Each time the CCU asks for an Script using its code, the Scripts module translate it into URBI commands that can be sent to the AIBO server. The Scripts module is only composed by three simple packages: *ScripsControl*, *ScriptsStoreRoom* and *RobotOutput*. Commonly with the rest of the modules it has a direct relationship with the CCU.

The package ***ScriptsControl*** contains the necessary methods to Exchange information with the CCU. ***RobotOutput*** is the intermediate layer between the Java commands and URBI commands. It is comprised by a set of methods that for each Java command give a URBI string to send to AIBO directly.

Package ***ScriptsStoreRoom*** is more complex and it contains the classes that are necessary to translate the information that gives the CCU. There are only three plus two text files: *dictionary, scriptInterpretor, functionInterpretor* and *emotionInterpretor*. The class ***Dictionary*** creates two different dictionaries using the two text files *scriptDict.txt* and *functionDict.txt*. Each text file has two columns of information separated by the character '='. What a dictionary does is to look for the data in the left part, and when it is found, it gives as response the data of the right part. Taking that into account the translation is quite simple using the two dictionaries. First the CCU sends to the Script Module the code of one script. *For example* *s001*. In this moment the ***ScriptsInterpretor*** has to translate that code into more defined commands. To do that creates an object ***dictionary*** using as argument the file *scriptDict.txt*. With that information, the interpretor cheks that file and substitutes its input (*s001*) for the output, that is the right part of the next sentence after the character '=':

```
        s001 = resetface + yes + end
```

The CCU receives those data so it has three different functions that have to be translated into URBI commands. To do that, CCU calls again *ScriptsControl*, but this time the interpretor is ***functionInterpretor*** that will use *functionDict.txt* in a similar way. The new dictionary has this line into the text file:

```
      resetface = Face.RESETALL();
```

The information that the ***functionInterpretor*** passes to the CCU is again the right part of the line, that is "Face.RESETALL()". This the name of a Java String defined in the class *Face* of the package *RobotOutput* that contains URBI commands. Looking to that in *Face.java* file:

```
public static final String RESETALL()
    {
        return "ledF1.val = 0 & ledF2.val = 0 & ledF3.val = 0 &
ledF4.val = 0 & ledF5.val = 0 & ledF6.val = 0 &" +"ledF6.val = 0 &
ledF7.val = 0 & ledF8.val = 0 & ledF9.val = 0 & ledF10.val = 0 &
ledF11.val = 0 &" +"ledF12.val = 0 & ledF13.val = 0 & ledF14.val = 0 &"
+"ledHC.val = 0 & ledHW.val = 0 &" +"ledWIFI.val = 0 & " + "modeR.val =
0 & modeG.val = 0 & modeB.val = 0";

       }
```

EmotionInterpretor class is called when an emotion has to be translated. So it uses the information that is passed from reasoning module through CCU to get the emotion and, depending on the type and intensity, the interpretor gives the URBI commands in the form of a String. With this CCU will send that information to the AIBO to show the emotion with the adequate intensity.

Then, the CCU sends that that String to the Connection Module. When the AIBO client sends the command is automatically translated into the string that it contains, so the information that the AIBO server receives is only a set of URBI commands.

5.9 Connection Module

This module is the responsible of managing the connection with AIBO Server, that is the AIBO robot itself. It has only one package called *ConnectionControl* that contains several classes. The relation between them is show in Figure 56.

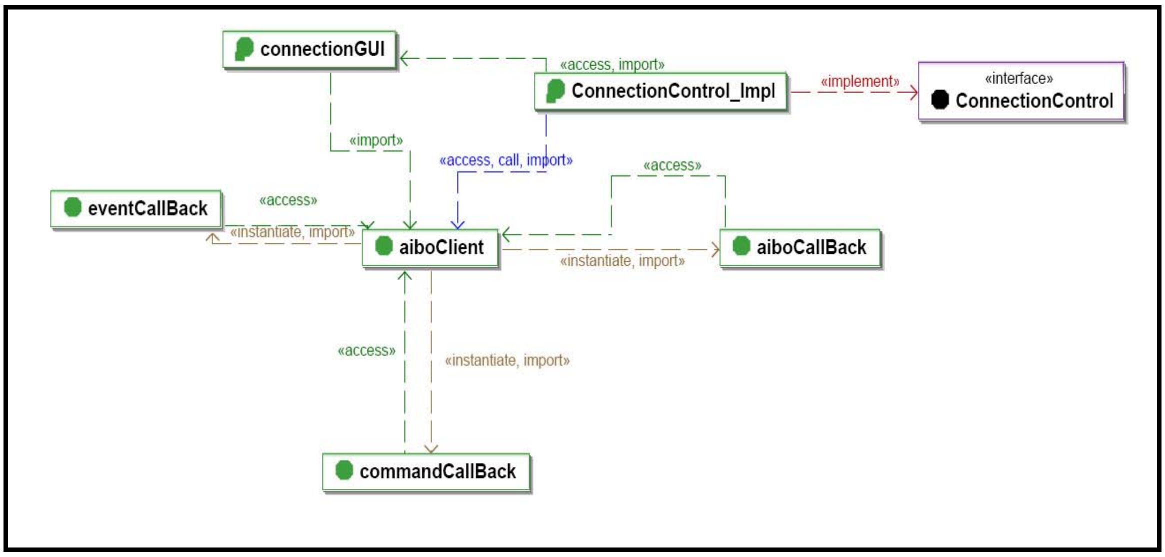


Figure 56: connection control class dependencies diagram

Similarly to the rest of the packages we can see that there is a *ConnectionControl* Interface and also its implementation (*ConnectionControl_Impl*). This manages the overall performance of the module and provides communication with the CCU.
When this module is initialized at the beginning, the control starts another graphical user interface, called *ConnectionGUI*. This gives the manual control of AIBO to the user instead of using the *Companion Mode* already described.

The *ConnectionControl* uses *aiboClient* to establish and control the connection directly with AIBO.

Let's see these two features more in detail.

5.9.1 AiboClient

The class *aiboClient* is the one that establish directly the connection with AIBO. CCU uses this class to send all the commands to AIBO server as well as to recover all the important information the server gives to the client. It uses the facilities provided by *liburbi* library as explained in previous sections.

This class is going to be the responsible of creating the connection and sending the commands (via Wireless LAN) to the URBI server (AIBO). This also has a nice feature using CallBacks. When the client sends a message with certain tag, the server will give the response of that message using again same tag. So creating some classes that uses the Callback features (*aiboCallback, commandCallback, eventCallback*) we can set some fixed actions depending on the server reaction. This featured is not used in this thesis, but it is very useful when the sensors of the AIBO work to recognize the environment.

5.9.2 Connection GUI

This GUI is used to both control manually the robot and to create the events that affect the reasoning process.

For controlling manually AIBO robot the Companion Mode should be active. Only in that case all the controls that are shown in the Figure 57 are available. The GUI lets the user connect to AIBO server with the IP address that appears in the main GUI. It also let the user make AIBO execute simple and complex actions (in the picture the names are visible), show predefined emotions and even introduce URBI commands directly in the text window.

For Companion Mode the rest of the features are available. There is a pane where the user can create all the events to try AIBO's behaviour. This is very useful because it makes the simulation much interactive than using predefined scenarios, and also gives the opportunity to introduce events in very critical moments to check if they give any problem.

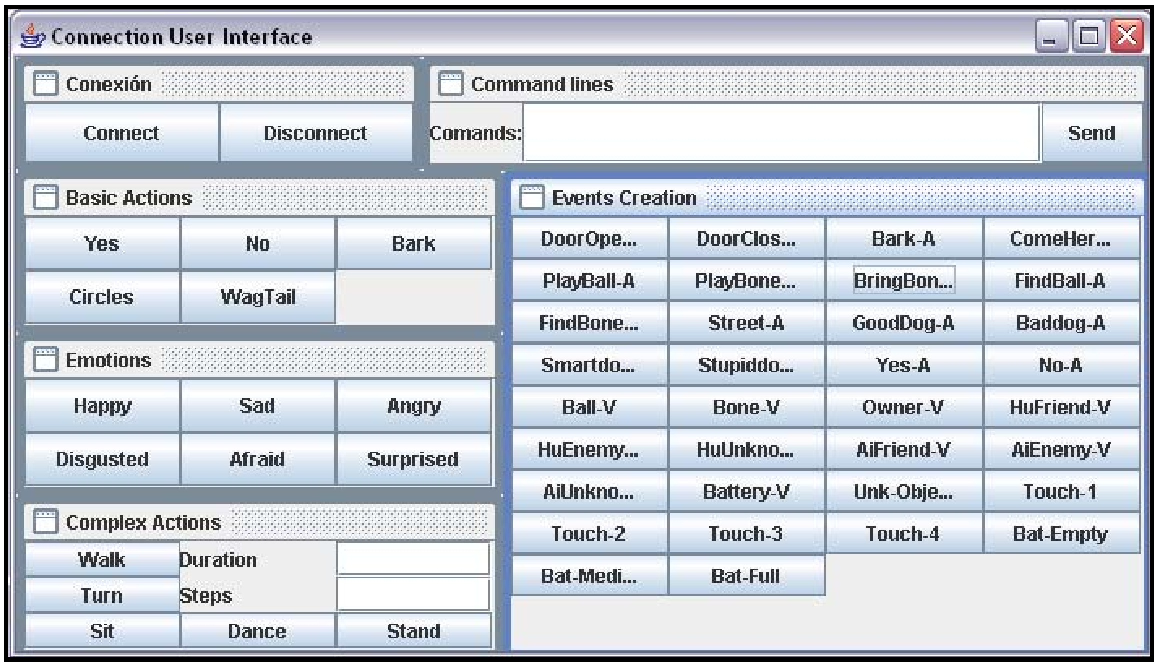


Figure 57: Connection GUI

6 Evaluation & Results

When we start this project we set the definition of the problem to solve: how to develop a companion dog with a personality model that let it influence the decision when some events occur around it. In addition it was necessary to test this model in a prototype, so finding or building a system that allows us to test the model was essential.

In the way of creating a real companion dog different personality models were studied and analyzed. After comparing their advantages and drawbacks, a new model based on needs was proposed. When analyzing which kind of system could be used to test our model in a real prototype we found that a new framework had to be built. The main goal of the new system to build was to make it powerful and flexible keeping it as simple as possible. We wanted to develop a computer system that allows future users to focus in the implementation of the mental-models (reasoning, planning, learning…) instead of building a new system from scratch that is the most time consuming task. So, in our case the aim was to get the appropriate balance between behaviour complexity based on the personality model and a powerful flexible implementation. Using that implementation a prototype was developed to test the validity of the model in a real robotic dog: AIBO.

In this stage the theoretical model has been implemented in a complete framework that allows us to work with the robot AIBO. So in this chapter the evaluation of the model, the implementation and the final prototype are done. The requirements that were already stated in section 3.4 are revisited along the whole chapter. The first section explains the development methodology and reviews the model created and the whole implementation process. Second section provides a summary of the features of the final prototype together with the requirements we achieved. The third section gives the technical evaluation, while the fourth compares our approach with the commercial model made by Sony. Some of the requirements have a subjective component for their evaluation, such as the level of realism of the prototype's behaviour. That is the reason why it has been necessary to set up an experiment with some viewers to evaluate those subjective features, using a survey. Fifth section exposes how the experiment has been carried out. The sixth and final section gives an analysis of the results of the experiment.

Chapter overview:

6.1 Development process

This section explains the development process technique from the analysis to the final implementation of the system. Tests have been made during the whole implementation process.

This project comprises mainly three different knowledge fields: psychology, man-machine interaction and software development. However the final prototype has to be able to embed the knowledge from the first two disciplines into a software product. That is the reason we turned to *software engineering* techniques to find a way to build the whole system. We chose a *V cycle development process* that it is shown in Figure 58. The model has been used as a guide for the development.

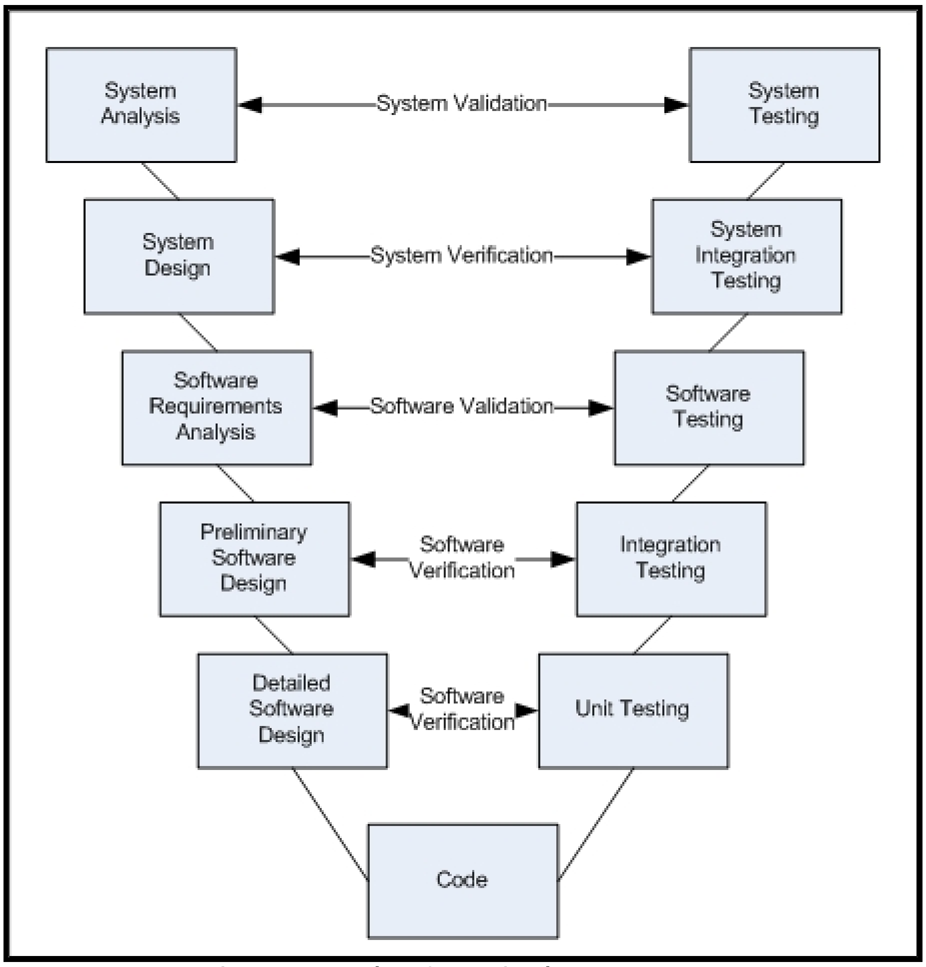


Figure 58: V cycle software development process

In the beginning, we started reading the documentation available to get a good background for the task to do. That concerns psychological information together with other approaches to social robots. Likely the document over what we settled the basis for our project was *Personality model for a Companion AIBO* [3]. In the start point, the first idea was to extend that work to create a complex companion dog. However, after a more careful analysis of that design, it was clear that it was not the right way to start. The main handicap was a combinational explosion due to the amount of variables in the core of the personality model. Thus we decided to create a new model (*Needs based Companion Dog*), based again in the *nPME* model but making some modifications to avoid that problem.

6.1.1 The model revisited

The proposed model uses as variables three different ***needs*** (*physiological, safety, love & belong*), ***personality***, ***mood*** and six ***emotions*** (*happiness, sadness, anger, fear, surprise* and *disgust*). In this case the whole model is based in the satisfaction of the needs, so the actions taken will be the results of evaluating the personality features, the current mood, the emotions and the level of the different needs (critic, medium or full). Since an implementation had to be made, some assumptions were taken to avoid that combinational explosion. Firstly we reduced the number of variables using the personality parameter: instead of continue using the *Big Five* traits of the personality, four fixed personalities were created (*friendly, unfriendly, aggressive* and *liar)*. Despite the fact that those did not cover the spectrum of real personalities, they provided us a good approach for the prototype implementation.[11] The mood was also reduced from a 2 dimensional space (arousal and valence axis) to 1D. So arousal varies from -100 to 0 for a negative valence and from 0 to 100 for a positive valence. Then, emotions, understood as external expressions of the internal mental state, are shown according to the current personality and the value of the mood. This implies that an aggressive personality will feel statistically more anger than a friendly personality for the same situations. Moreover, personality influences the mood, and the arousal value of the mood gives a clue for the resulting intensity of emotion.

This model also assumes that has to perform in the real world. Here some events occur that define a certain situation, and here is where the system has to reason and act in some way. The personality model in these cases will influence directly the election of one or another way of acting when facing certain events. Moreover, the election of one certain way of execution will affect the personality parameters (nPME) what has two advantages. First, this assures a continue flow of actions where the personality parameters are changing according to its past, so these variables moves usually in a continuous manner.[12] Second advantage is the possibility of learning. More in detail: the personality influences incoming actions and these actions will influence personality parameters and also future events. This information can be used for a learning system to evaluate the set of actions that give a better position in the nPME parameters (higher level in needs, high-positive values of mood and positive emotions) against the situations where those parameters experience negative values, and use

[11] It is necessary to point out that the robot we use for the test has limited expression capabilities and this was taken into account during the whole developing process. Due to these limited expression capabilities we assumed that a reduction in the number of variables would let us enough complexity reduction for the implementation of the model and, at the same time, it remains enough expressivity for the actions and emotion to face the experiment.

[12] There are some events that can change radically the flow of actions. For example if the dog is playing happily but an enemy appears. This may result in an abrupt modification of the needs, mood and emotional parameters to face that situation in the best way.

this information to make the system choose the actions that lead it to a better situation. Actually the model is always looking for the satisfaction of the needs. This can be seen from the perspective of a search process where local maximums of needs satisfactions can be achieved. If there is an event that makes these levels worse, the system will try again to look for another local maximum performing a set of actions.

6.1.2 The implementation

Once the model was finished, the requirements of the new system were carefully extracted according to our goals. In that point we start to create iteratively some designs for the architecture of the system, resulting in the one shown previously in Figure 38.

Then each module was carefully analyzed and designed, trying to achieve the maximum independency between each pair of modules. The objective was to make every module independent except with the CCU that would act as an intermediate in every information exchange as well as the main control of the whole system.

It is important to mention again, that in spite of the fact that this is a project where the psychological model is the most important issue, we had to materialize this model in a real prototype. We found that there was not another useful available framework to test this model, so necessarily this system was designed and built it from scratch. The main idea during the design stage, as said above, was to keep the independence of the modules to create a flexible system. This was necessary for our prototype, but only partially. In fact, the main motivation to develop such a complex system was to provide future researches with a framework that can be reused. Actually the implementation of our system has been the most time consuming task of our project, but hopefully with our effort we will avoid much work to new researches, so they will be able to focus mainly in the psychological model.

After all the modules were designed, the coding part started. We were building module by module. Every time a module was implemented some tests were applied to check its correctness. If a mistake was found, it was solved before starting with a new module. After the first module was implemented and checked (*events-module*), an initial CCU was also built. It was essential to start testing the modules together with the CCU and the previous modules, as can be seen because of the fact that the reasoning module needed the events to start working. So, after a module was tested isolately, it was tested again in combination with the CCU and the previous modules. This way of working made the CCU to grow up iteratively in each step, increasing its communication utilities. This let us to have a close control of its advantages and drawbacks, to improve the last ones as soon as they were discovered.

When the whole system was implemented, several tests were made to assure its correctness. Reasoning model was tested more carefully. It is obvious that the final result will rest over the correctness of its reasoning procedures.

In this moment the system accounted five different modules:

- ***Central Control Unit module***, that controls the system and acts as an intermediary to exchange data between the modules. It has one GUI to start the automatic mode where the personality model influences the actions.

- ***Events module***, to create and destroy every external event that can occur during execution (e.g.: owner image, dog sounds, battery events, etc). In this module timing tasks were crucial, because they set the amount of time that a certain event can affect the reasoning process.
- ***Reasoning module***, uses the events, last executed actions and internal emotional state to inference the emotion expression to show, the script to choose and the change in the variables of personality.
- ***Scripts Control module,*** stores and translate the different scripts and actions made to a language that the robot can understand. It includes the basic actions in URBI wrapping in Java commands, and the more complex actions as a result of different combinations of the basic ones.
- ***Connection Control module***, manages the connection with the robot and sends all the commands to execute the actions chosen by the reasoning module.

When the whole system was running alone, we made the integration with the real robot. AIBO was the robot chosen that can communicate with our system via wireless. Many adjustments had to be made to have it working in conjunction with the rest of the software.

Last stage in the development has been the 'System Testing' where a criterion as natural and continuous behaviour or event-emotion-action coherence was taking by the author to create the whole complex behaviour of the system. Since these are quite subjective criteria, we developed an experiment to test whether those criteria in the way we implemented were shared by more viewers.

6.2 The prototype

In this section we summarize the final features of the prototype that is already implemented in relation to the requirements that have been accomplished (these will be shown in bold letters between brackets). To review the list of requirements, check point 3.4.

The robot used for the prototype is AIBO made by Sony that can be controlled using different programming languages. Among them, URBI is a popular universal platform for different robot architectures (**Req. 1**). The robot has visual, audio and touch sensors among other, and a complete set of effectors to move head, neck, paws, tail, ears and mouth. Everything is managed by a 64-bit RISC processor (**Req. 2**). Java has been used to implement the framework that supports the physical control of the robot and the reasoning module (**Req. 11**). Java and URBI can be downloaded and used for free (**Req.3**) and are together with JESS well-known programming languages for many developers (**Req. 6**). The reasoning system follows the model of the BDI agent where beliefs are coded in the way of events and inference rules, desires are identified with the satisfaction of the needs, intentions are the need that is going to be satisfied executing some scripts and plans are coded into those scripts. (**Req. 5**). The reasoning has been implemented as an expert system, a well known AI technique (**Req. 7**). The whole system has been designed to act in a real environment and it performances in real time (**Req. 9**, this will be further evaluated during the experiments) where a minimum of ten different threads are always running

(**Req. 10**). It has been conceived also as the basis for future research, so the ease change of the components has been essential (**Req. 8**) although this has not put into practice. In addition, all the system will be kept as an open-source code for further extensions, and a big effort was also done to make it easily understandable. To give and idea, with a line counter we found that the number of comment lines is around 3200 for a code of around 7200 lines. So approximately, 1/3 of the number of lines written for our system is comments - excluding exclude Jess files- (**Req. 4**). We add Table 17: Numerical implementation statisticsTable 17 to show some numerical data about the implementation.

Table 17: Numerical implementation statistics

Some Code Statistics	
Number of Java files (including tests)	**92**
Number of total lines in Java files	**11680**
Number of code lines in Java files	**7217**
Number of comment lines in Java files	**3211**
Number of Jess files	**3**
Number of rules for needs	**38**
Number of rules for personality (average)	**98**
Number of txt files for configuration (needs, scripts, etc)	**3**
Number of basic URBI functions wrapped in Java	**161**
Number of complex functions (functionDict.txt)	**149**
Number of scripts (scriptsDict.txt)	**66**

Following with the rest of requirements of Table 5, we developed a system that had to gather some features to provide the user with a realistic dog experience. Although our main objective was to focus only in the reasoning module, we found that the output of the robot (movements, sounds...) conditioned a lot the user experience. So it was necessary to complete the set of actions from *RobotOutput* package [3] (**Req. 22**) to provide a rich behaviour and a more complete experience. The number of complex functions and the scripts in Table 17 can give an idea about the amount of actions that robot AIBO can performance. We also added around 30 sounds more apart form the ten that URBI server uses as default (**Req. 12**). In addition we wanted to make AIBO as a domestic robot, so ease of use was a main goal to pursue. In that sense we created a GUI that can manage AIBO only with one button -"Companion Mode" turns on the automatic behaviour- (**Req. 15, 17**), but it is also true that the GUI's that are currently running have been developed more for scientific and testing purposes than for a possible final user. Besides the events that AIBO can sense have to be fired from one of the GUI's so that diminishes the interaction experience. Further improvements have to be done on this.

Focusing on the psychological model, implemented in the reasoning system, we have seen that the model is based on needs, where each need acts a 'bucket model' that empties during time or because of certain events –amygdale approach (**Req. 29**)- and the behaviour of the dog aims to fill those buckets (**Req. 25**) satisfying the needs (**Req. 28**). The mood was reduced to one axe (**Req.26**), and we have created for the prototype 2 different personalities –friendly and unfriendly (**Req. 24**) – reducing the total number of variables for reasoning and therefore restraining the combinational explosion in the number of rules. Many scripts have been implemented to show different behaviours that are executed according to the situation, personality and the rest of variables of the mental model (**Req. 27**). The events that provide information to the reasoning about the external world cannot be sensed directly by AIBO sensors, but those events are wrapped using a GUI where each button matches one certain event, so when the user controls the GUI AIBO can know some important information that conditions sometimes its behaviour (**Req. 21**).

Besides these features, the reasoning system was conceived to be easily adaptable and modifiable for other purposes. Some psychological experiments carried out with AIBO showed that people with social disabilities felt more open after interacting with AIBO. Moreover AIBO (or other future robotic dogs) can watch elderly people or children providing information to a nurse or to their parents about their state. This is a promising feature, so it is very important to keep the adaptability spirit alive, letting future applications. We have done it using modules integrated by expert system and 'txt' files to define each script. First, although some technical background is needed to adapt the code to a new behaviour, the expert systems allow a user to change their rules in a very easy way. We also translated the theoretical model into code but without hiding the psychology model behind difficult programming commands. That was a difficult task, but if the reader takes a look to one of the rules of the second inference engine (*friendly.clp*), for example, he will practically understand how it works even without technical knowledge. The basic concept to know is that when the left part of rules (before '=>") is true, the system will perform the right part (after '=>'). More information about this is available in Appendixes C (9.3). For example we can get next chunk of the code. It is the first rule of the 'friendly' personality to satisfy need 1 (physiological). Some extra comments (red) were added to ease the reading task.

```
;;First rule to satisfy its need. It feels tired.
;; Emotions: Happy(1), Surprise(6), Sad(2), Angry(3), Afraid(4), Disgusted(5)

(defrule n1-1-Pos-1          ;      X need, Y order,script, M mood. Name of the rule
       (LEFT PART)
   (declare (salience 100)) ; States the priority of this rule among the others
    ?a<-(lastNeed physio) ; The need that the first engine has chosen is physio (need 1)
    ?m<-(mood (valence ?val)(arousal ?ar)) ;Mood Gets the current mood parameters
    (test (= ?val "Positive"))       ; If valence is positive…
    ?e<-(emotion (type ?type))       ;Emotion Gets emotion parameters
    (not (alreadyExecuted "Physiological" "s001p" ))  ; If this rule hasn't been executed
    (not  (alreadyExecuted  "Physiological"  "s001n"  ))    ;  If  rule  's001n'  hasn't  been
executed

    =>
       (RIGHT PART)
    (printout t " ///[friendly.clp] SCRIPT: n1-s001 - p - surprised" crlf) ; Control data
    (call ?*control* decPhysio 2)     ; Decrease need 1 two points.
    (modify ?e (type 6) (intensity ?ar)) ;Modify emotion 6
    (modify ?m (inc -10)) ;Arousal modifies also valence ; Modify current mood
    (retract ?a) ;Remove a fact that was checked as true in the left part
    (assert (script "s001p")) ; Script 's001p' has to be executed
)
```

Second advantage of the system, as we pointed out above, is the use of 'txt' files to define actions and scripts. A combination of primitive actions (taken form *RobotOutput* package) to make a complex one can be set in the file *functionDict.txt*. Every line defines a new function that can be used by a script. This line comprises a name of the function and then a combination of the primitive actions. For the scripts a file called *scriptDict.txt* is used. In every line there is a different script. It is formed by the name of the script and a set of functions whose names are taken from *functionDict.txt*. During the development and testing stage of the final prototype, these files have become essential and it has been checked that they constitute a very good way to develop, test and improve quickly.

6.3 Technical evaluation

To make the first evaluation of the prototype we implemented two different personalities that will act in several different situations. These situations have been organized according to the need they are related to. For example, situation when somebody new enters at home has been classified as a situation that specially affects second need, safety, even if the new character starts to play (that can be related to third need). The aim was to design simple situations, that allowed continuous behaviour but without creating only one linear way of performing the different actions. That is why we use some 'finite-state' diagrams to show the different situations that friendly and unfriendly AIBO will face. These diagrams can be found in Appendix B (9.2). Thanks to them we can show that there are several ways to face the satisfaction of a need, and at the same time, when satisfying one, there is a small set of states that are reached in a sequence. Each state represents one of four scripts to execute (depending on personality -2- and mood -2-). So each script represents instead of a complex action, a certain situation.

A very brief classification of the situations we created, relating to one or another need is shown in Table 18. A situation can be seen as combination of current events, past events, personality, past actions and current mood. Emotions also evolve in each situation but they were not used to classify them.

Table 18: Situations implemented for AIBO according to needs

Need 1 – Physiological (battery, food, rest,…)	
Aibo is hungry	Aibo finds the food
Aibo does not find the food	Aibo eats
Aibo sleeps	Aibo switches off
Need 2 – Safety needs (new people/aibo's/objects)	
Aibo looks for somebody	Aibo finds a friend (Aibo or human)
Aibo finds an enemy (Aibo or human)	Aibo finds an unknown character (Aibo or human)
Aibo plays with the character	Aibo flee from character
Aibo attacks the character	Aibo hides from the character
Aibo finds an unknown object	Aibo finds its ball/bone
Need 3 – Love & Belonging Needs (with its owner)	
Wander	Look for owner
Waiting bored	Owner appears
Owner gives a command	Ownre strokes aibo

The experiments carried out to check the liability of our system were designed to evaluate aspects of the implemented model such as the variation of the different needs, the change of situation when an event occurs, the correctness execution of a script, the evolution of the emotions, etc.

The methodology of these experiments can be divided in two stages:

1. The execution of every possible way for each need. This implies the set of specified initial values for each need. Then, for a need, some initial states need or not

triggering events. So for test all the possibilities, we did it with those events and without them.

2. The interleaving execution of different needs. For this we made different combinations in the order in which the needs were executed (123, 132, 213… with some repetitions). The events chosen were selected randomly.

All the experiments were carried out using AIBO robot, and some variables that set times were adjusted to get an appropriate performance. Since the events are not recognized directly by AIBO, the connection GUI was the tool used to trigger each event.

RESULTS

I. *Needs evolution*

The behaviour of the **need** levels were as expected. Their values decrease certain amounts during time correctly. When especial events occurred ('battery empty event', 'enemy image' for example), the needs were automatically updated (need 1 decreases and need 2 decreases respectively). Needs also could increase their values only with certain events that were not related directly with the satisfaction of the need in the description given. For example, owner command 'come here', related to need 3 (love), increases need 2 (safety) to give more chance of the situations in need 3 be executed. Technically the system performs as expected, according with the values we set. And, finally, the needs can increase its value due to the execution of scripts that satisfy them. We also tested this for several different situations and the need that was being satisfied increase its value. This was one of the main features of the system and it works successfully.

The change of the order of needs execution also works as expected. If battery is empty, need 1 is the one to satisfy. When need 1 or 2 reach certain level, the need changes to an upper need (if lower needs have enough values). If in a upper need there is an event that decrements the level of a lower need, the system makes that change automatically to satisfy the most priority need. The list of the transition of the needs according to their values can be found in Appendix G (9.7). So the evolution of the needs have been properly accomplished.

II. *Events consequences*

When the events occur, as commented above, sometimes (depending on the event), it produces a need change. But also, those events provoke different ways of execution. For example, if Aibo is looking for its owner and he cannot find him, the system will make him execute the states where Aibo shows disappointment and then it has to wait bored until a new event happens (this event can be the owner image, a command, etc. but also a battery event, an enemy that enters into home, etc.). If when Aibo is looking for its owner, he can find him, Aibo will react appropriately attracting his attention and waiting for some 'love' demonstration: stroke, commands, etc… The test shows that the change in states according to the events works as expected. So the changes in the actions sequence according to the situation have been successfully accomplished.

III. *Scripts execution*

When a script has been chose to execute, the system sends it to the robot server to be executed physically by Aibo. This execution is often correct. This execution seldom gives problems, but those problems are related with the real level of battery in Aibo during tests or with the wireless connection, but not with the way of working of the system. So the translation and delivery of the actions from the scripts in the appropriate situations are also accomplished.

However the experiments have shown a possible shortcoming in the implementation approach. When an event occurs, the event has a timer that sets the influence time of that event. When the timer is finished, the event is properly removed from memory and also from the reasoning system, so it will not influence Aibo's behaviour any more. Until that point, everything worked as expected in the design. The CCU controls the main time line of the system, while the needs and events have their own time line. This makes that one script has to be completely executed by Aibo to let the CCU do another iteration to evaluate needs and situation again. This implies that the number of actions for Aibo in each script will condition its time response when an event occurs. Besides, if the influence time of an event has not been properly set, according to those actions, the event can be lost and removed from memory before Aibo can react to it. As said, the cause is the number of actions in each script, and the influence time set for each event. So this shortcoming can be overcome (keeping the integrity of the CCU model, that continues being useful for researching purposes) keeping the scripts short (so, for the same level of quality in the actions, more states must be defined). In addition, the creation of many new short-states (and therefore many new short scripts) will improve the reusability of the scripts and increase the complexity of behaviour.

IV. *Emotions evolution*

The evolution of emotions works as expected in the implementation stage. Each emotion has five levels of intensity. The lowest level is associated only with some figures in the face leds, while the highest level combines leds figures, movements of neck, head, mouth and tail and also a characteristic sound. The level of emotions is calculated using previous level of mood, what gives continuity in the increase or decrease of emotions in certain situations. Mood also influences the emotions because a negative mood will imply zero intensity for positive emotions, and a positive mood will provoke the same for negative emotions. This has sense in the most of the cases where Aibo is very happy, and a change on the situation makes him feel angry. In these cases the level of anger must be minimum until certain time has reached. So Aibo counts with *emotional-memory*. However, despite the fact that in some cases this emotional evolution gives great results (Aibo is waiting bored, and its sadness increases during time until the highest sad level), there are other changes of situations that need a sharp change in the emotion. For example, when a real dog is very sad because its owner has not been at home for long, the mere fact that its owner enters home will make the dog be absolutely happy (maximum level). The system as implemented makes possible to remove the evolution of feelings when necessary just by setting manually the increment of some specified emotion. This has been tested in the situation when Aibo encounters an enemy, and it works successfully.

V. *Personality behaviour*

The way Aibo behaves according to one or another personality has been tested using two personalities: friendly and unfriendly. We implemented one case in our situation scheme where this could be perceptible, only to check that technically the election of personality gave different behaviour. A different set of scripts was created and a new file *unfriendly.clp* was written. Actually we used as a basis the file *friendly.clp,* introducing some changes in some rules.

In the case enemy enters home, that can be easily perceptible that a friendly Aibo, as programmed, will be afraid of the threat, trying to flee, while a unfriendly Aibo will feel more aggressive and it will express very explicitly its anger when it is avoiding the entrance of that enemy.

6.4 Comparison between Needs Based Companion Dog and Sony AIBO Mind Software

To make a more in deep evaluation of our model and prototype, a comparison between the model developed by Sony *AIBO Mind software* and our model *Needs based companion dog* is given. This comparison has been made using the "Emotional AIBO" [34], a research the Man Machine Interaction department of TU Delft using Sony mind's commercial model in an AIBO ERS-7.

That research was made under the constraint that Sony Mind's model source was not open, and there was not complete technical information about that. It was only available the user instructions manual. Therefore the research was made by using AIBO during seven hours. These results, despite the fact that cannot be conclusive, can give some idea of the final results that the prototype achieved in a commercial approach.

Table 19: Comparison between AIBO Mind Software and Needs-Based Companion Dog

AIBO Mind Software	Needs-Based companion dog
Maturation feature	Two different **personalities** already implemented. The framework allows the inclusion of a module for a maturation process based in learning.
Sound and visual recognition	Made by using a Graphical User Interface
Five **instincts**: love, curiosity, movement, hunger (low battery) and sleep.	Three **Needs**: physiological (hunger, battery, rest, etc), Safety (curiosity, surveillance instinct, watch-dog, etc), Love & Belonging (love, games, owner attention, etc)

Six **emotions**: happiness, sadness, anger, surprise, fear and dislike.	Six **emotions**: happiness, sadness, anger, surprise, fear and disgust.
No clear patterns in lighting face leds.	Every pattern show one defined emotion [3]
User interaction can be classified as positive (encouragement), negative (scolding) and neutral (commands).	**User interaction** can be classified as positive (encouragement, certain commands) or negative (scolding, ignoring) depending on the **mood.**
Positive and negative behaviours do not seem to be the representation of the different intensity AIBO's emotional states.	5 different levels for the intensity of each emotion.
Positive and negative interactions with user always trigger corresponding emotional responses in AIBO, irrespective of its instinctive state.	The increase of decrease of the emotion in every state depends on the need. Satisfaction of need of gives one gives lower happiness increase than satisfaction of need three.
AIBO does not seem to have an on-going emotional state.	Emotional evolution through **mood** variable.
Emotional behaviours without user interaction (linked to instinctive states) seem to display at random.	Emotional behaviours always corresponds to a defined situation with meaning (inside its need satisfaction).
Patterns difficult to recognise.	Realistic sounds for ease of recognition.
Interaction affects emotional state	Interaction affects mood and emotional state.
Interaction does not affect the overall instinctive state.	Since Love & Belonging is considered a need, interaction does affect overall needs state.
AIBO will react always to encouragement or scolding by the user.	AIBO react to encouragement of scolding only when its third need must be satisfy. Those events change some needs values to prioritize relatively the third need.
When AIBO is hungry, it will ignore the rest of commands or interactions except user encouragement or scolding.	When AIBO is hungry, it will ignore the rest of commands or interactions. (This can be modified easily if a 'safety' event must be prioritized.)
AIBO cards have priority over all other instincts or voice commands.	No AIBO cards.
AIBO will only sometimes react to the voice commands, depending on its instincts.	AIBO will only sometimes react to the voice commands (GUI), depending on its needs.

To give a better idea of the difficulty in Sony's model to identify emotions, we tried to get pictures of AIBO happy and anger. The experiment took one hour where the first half we were encourage AIBO and the next half we scolded. The same patterns were repeated in both halfs, where some pictures were taken. Due to the impossibility of identify clearly the emotions we could not organize them. It was even impossible to get a picture of AIBO feeling angry because the different expressions are shown too fast to give a real impression of a defined emotion. The movements equally were similar in both halfs and the same happened with the most of the sounds. Besides those sounds are not real, they are all like 'electronic' beeps, so it does not help for the identification. We included some pictures of those apparently random patterns in Figure 59.

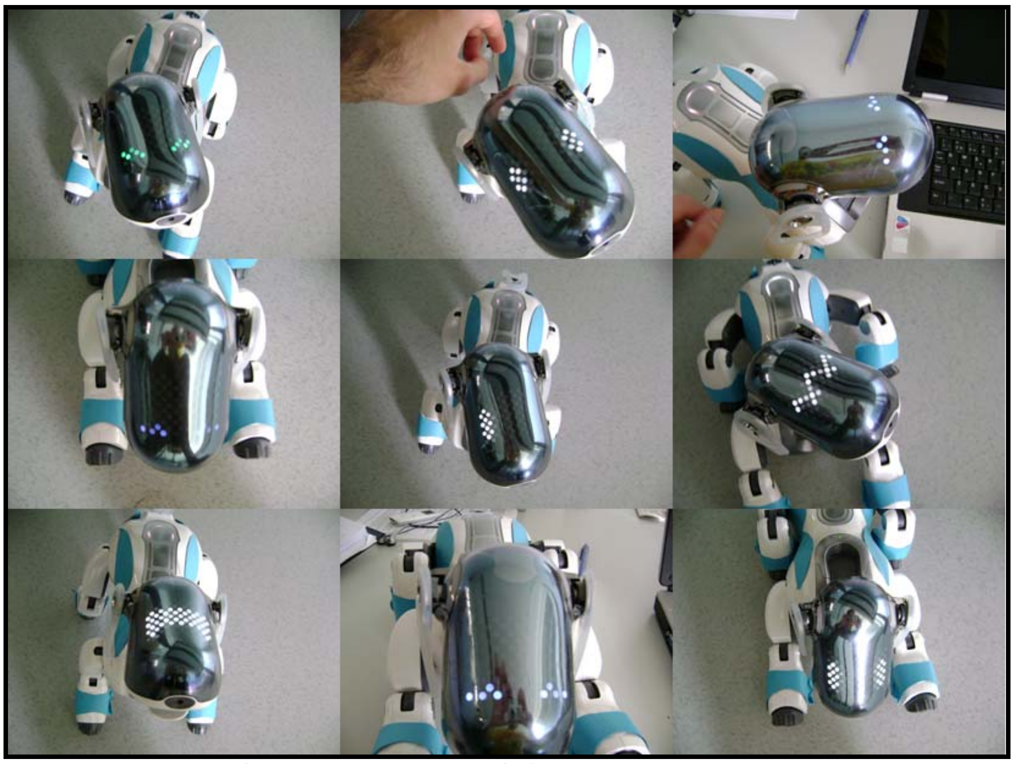


Figure 59: Several Sony Mind software AIBO patterns

In comparison, we also included the different 6 emotions that were developed in *Peronality model for a Companion Aibo* [3]. These are well defined although the identification experiments shown that it is very complex show clearly 6 emotions using only the leds patterns.

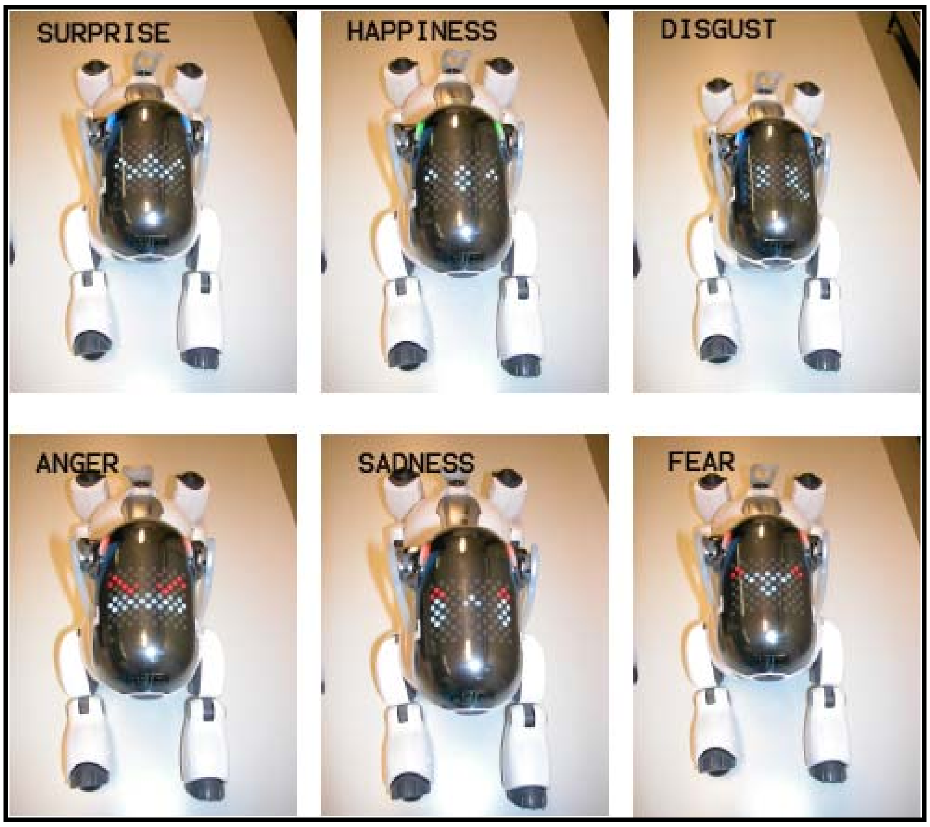


Figure 60: Emotion expressions developed in Personality model for a companion AIBO

In *Needs Based Companion Dog* we used the patterns of Figure 60, but we included five different intensities. To show those intensities we used the intensity levels of the leds in combination with head, neck, tail and mouth movements. The emotional experience was increased with sounds recorded from real dogs. Despite the fact the sound cannot be included in the report, we also include a set of pictures that show the different intensity levels of the emotion <u>sadness</u>, to give an idea of the increase of the emotional complexity in this new model.

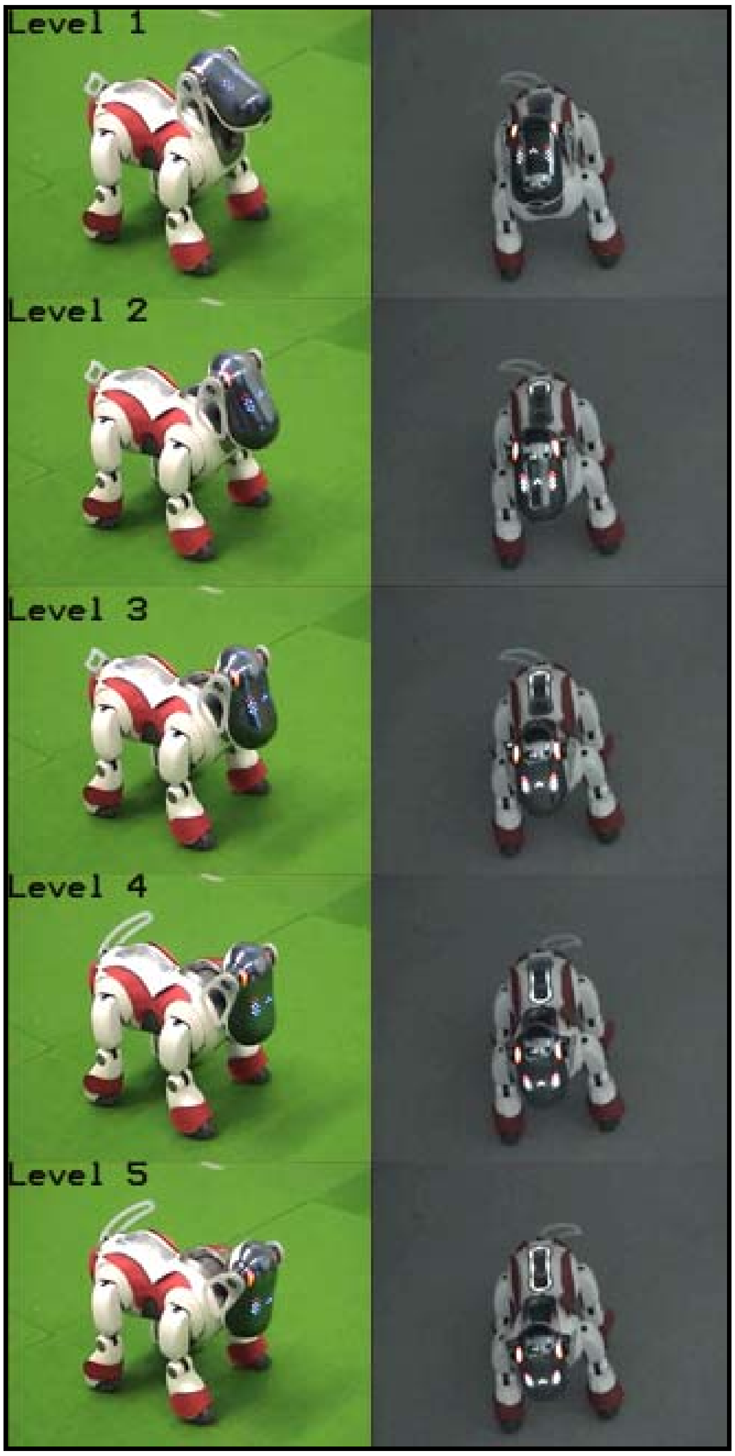


Figure 61: Intentisty levels on Sadness emotion (Needs Based Companion Dog)

6.5 Experiment set-up

This section explains which of the requirements need further evaluation. It also explains the experiment we use to give more data about subjective impressions when watching AIBO behaviour, and describes the four different scenarios that were used for the experiment.

If the reader checks the whole list of requirements shown in Table 5 and marks the ones that have being accomplished according to the paragraphs of last section, she or he will notice that there are some of them that have not been mentioned. The reason is that they need another type of analysis due to their subjective nature. We review those requirements in Table 20.

Table 20: Requirements with a subjetive component in their evaluation

Requirements to test in the experiment	
Req	**Description**
9	**Real-time system:** it has to react in an appropriate time to events in the world.
13	**Companion robot:** behaviour similar to a normal companion pet.
14	**Continuous behaviour:** the sequences of actions must be continuous without annoying stops between them.
16	**Suitable for children:** natural and predictable behaviour. No unpleasant actions.
18	**Personal behaviour:** personality defines different behaviours in same situations.
19	**Continuous emotional state:** emotional states have to evolve slowly in the most of the cases.
20	**Coherent reactions:** actions should be related to both internal state and environment.

To test if those requirements have been successfully accomplished in our prototype we stated a set of aims that our prototype must achieved:

1. AIBO must response to some events with a change in behaviour and emotions. This change must be done before the event was removed and as natural as possible (Req.9,14).
2. The actions in the environment must be inspired by the behaviour of a real dog (Req. 13).
3. The behaviour must be adequate for children like a puppy, even if the robot appears to be angry. (Req. 16)
4. The change of personality will change the actions and emotions for similar situations. (Req. 18)
5. The emotional evolution should not be sharp but should do recognizable. (Req. 19)
6. The actions performed by the AIBO should be logical according to the situation where he encounters. (Req. 20)

To check if those aims were achieved, we have built a group of four different scenarios that can be seen as a continuous situation where different events occur. The events can change eventually the execution flow of actions and emotions. Although the fact these scenarios can be seen as a continuous scene, the time gap between them is not realistic at all to improve the experience of a continuous behaviour. The scenarios are based in a continuous execution of the different states shown in Appendix B (9.2).

Due to the lack of a navigation system in AIBO's reasoning system the actions of walking and turning have been chosen according to every situation (every state of the finite-state diagrams) and according to the real scenario we mounted in the laboratory. The rest of the actions were chosen to get a 'realistic-dog experience'. The emotion expressions were chosen for every script and personality but its intensity continued being controlled by the mood variable, as explained in the model. However, according to C. Bartneck [25], usually it is a developer's task the mapping of the emotional state of the character into its behaviour. Therefore, this mapping is often very influenced by the own perception of the developer. That is the reason why he suggests the figure of a *personality designer* and also an iterative process to improve the relationship between the behaviour and the emotional mental model. In our case we did not have access to a dog ethology expert to aid in the personality development, but we did follow the idea of the iterative process. This was carried out asking five different students of this faculty during the implementation process about their subjective opinion when they saw AIBO's behaviour. The author used that information to improve the robot's movements, what resulted in an increase of realism.

Once that stage was finished, the experiment with the four scenarios was carried out. Some videos were recorded during those. The behaviour of the dog according to its internal variables is described and analyzed as way of increase the information given in section 6.3. When these scenarios were tested in the lab, and the videos were, we carried out a survey with 20 students. We presented the videos to them to gather some data about their subjective impressions about AIBO's behaviour.

As said above, for the experiment there are four different scenarios. In the first two scenarios, AIBO is hungry, what means that need 1 is in a critic state. Since need 1 has the highest priority AIBO should satisfy it. If the need can be satisfied, the most priority need will change and AIBO has to act in consequence. Otherwise, as need 1 is physiological and represents that it is hungry (or in a robot, that its battery goes off), it has not enough strength to performance more actions. Third scenario starts with AIBO looking for its owner. He feels lonely (need 3 critic) so it will look for a way to satisfy it. Later, AIBO finds its owner and starts playing. In the fourth scenario, AIBO finds an enemy, so need 2 (safety) decreases immediately. It must react in consequence. In Table 21 the specification of the test environment where the scenarios will be recreated are given. Two pictures are included to show the test environment where AIBO performances during the experiment in Figure 62.

Table 21: Test environment specification

Test Environment	Specifications
Robot	AIBO
Place	EWI Room
Objects	Walls, Door
Special Objects/Humans	Food tray, Food, Owner, Enemy
External PC (control)	Laptop Toshiba Centrino with wireless

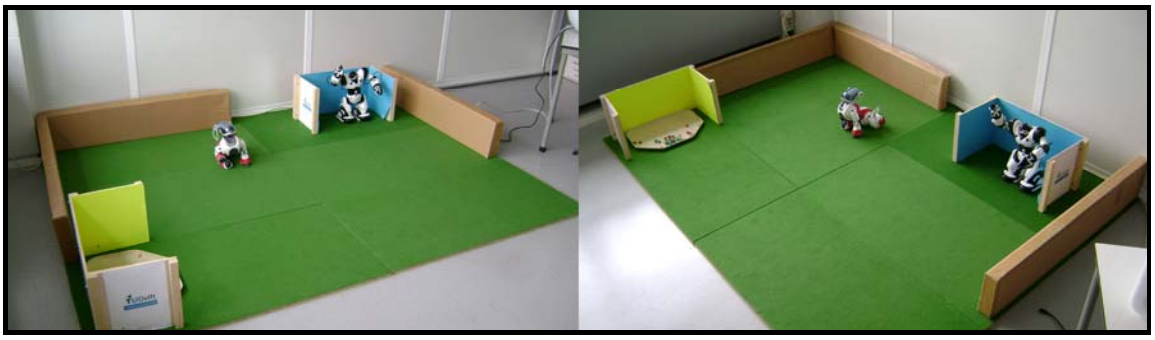


Figure 62: AIBO in the environment for the experiment

6.5.1 Scenario 1: AIBO is hungry but there is no food

In this scenario AIBO starts feeling hungry. This sensation can be translated into a low battery event in the robotic world. In the system, physiological needs are the most priority ones. This means that the surveillance of the robot is the most important task to care of. As a consequence AIBO would leave every other task if the level of its need 1 is critic.

In this case AIBO looks for food, but this can not be detected what has two consequences: first, its emotional state decreases the value of the positive mood and therefore positive emotions and increases the weight of the negative ones (sadness, anger…), so according to its previous state AIBO will show an emotional reaction. Second, since AIBO could not find the battery, he cannot eat and therefore its security is threatened. So the best option is to take a low-energy action, like sleeping.

Experiment results:

Aibo acted as expected. He starts in a neutral emotional position (0 intensity for every emotion). When he discovers he has not battery, feels surprised, but it is not a good situation for him, so he becomes sad. He tries to find the food but he cannot, so he feels disappointed and, sadder because its mood decreases. In addition the battery is going off so he decides to sleep for a while. When the battery reaches a very critic level, the energy consumption of AIBO is reduced to the minimum, and no expression is shown. In this way AIBO has more possibilities to survive, keeping the performance of the battery in the minimum. Logically this will not increases the value of need one, but at least speeds down the loss of battery and therefore the decrease of this need.

6.5.2 Scenario 2: AIBO is hungry and there is food

The situation starts when AIBO is sleeping. Its owner puts some food in the tray and calls AIBO. Thanks to this event AIBO reacts getting up, and goes to the food tray. He eats for a while until the batteries are more charged. The satisfaction of the need must show an increase of the mood and therefore one of the positive emotions.

Experiment results:

AIBO behaved as expected. He gets up with its leds off. This is because during the lowest battery stage the emotions are set to zero automatically to save battery. When he goes to the food and starts eating, the mood value increases giving the chance to show higher levels of positive emotions. The action of eating is repeated until the battery reaches certain level meaning that he has energy to continue with its normal life. Every time AIBO eats the mood increases a certain amount and the level of happiness also increases with it. So at the end AIBO feels happy and the level reached is enough to show it. Moreover, the need 1 has been successfully satisfied.

6.5.3 Scenario 3: AIBO is bored and needs someone to play

When AIBO finishes eating, need 1 is satisfied, need 2 continues in a high value but need 3 is quite low. So according to the table in Appendix G, need 3 must be satisfied. If this happens means that AIBO needs love from its owner. So AIBO should look him/her. If AIBO cannot perceive its owner, he will feel sader as time goes by, with the mood decreasing. If he finds it, AIBO will react positively and mood increase and also the positive emotions. Depending on the events the owner provoke (its own image, strokes, commands) AIBO will react increasing its mood, so the positive emotions will be shown stronger.

Experiment results:

At the beginning we do not give the event of the owner image to AIBO, so he starts looking for him/her, but no result. This makes AIBO feel disappointed and the level of sadness increases. During time AIBO moves its head down showing the different levels of sadness. Then, the owner comes, so that event is introduced in the system. Emotion changes to positive but mood has a very low level because of the time he was waiting, so the intensity of the emotion is zero at the beginning, although the mood increases a bit. The owner strokes AIBO and says that he is a good dog. These events get a higher value to the mood so at the end AIBO has reached a positive mood and happiness is expressed.

In reality, the most of the dogs, in spite of the fact they are very bored and feel lonely, when their owner enters at home, this emotion changes completely to the highest value of happiness. So a more higher change may operate in the mood to reach that level with only one event (at least with the owner image event in these cases). In the system this can be done quite easily due to the fact each pair of rules define a situation, so the value of the emotion can be change directly or give a higher increment continue using the mood (this gives worse results but seems to be more correct in the logic of the emotional evolution).

6.5.4 Scenario 4: AIBO is playing but an enemy enters home

When AIBO is playing with its owner, suddenly he hears a strange sound. Behind the door he will find an enemy image. This should make need 2 (safety) decreases quite a lot. If this happens, AIBO should react appropriately looking for a set of actions that makes him increase that need. In this case we introduce a change in the behaviour of the *friendly* and *unfriendly* personalities to appreciate the different actions. The first one will flee when he faces the enemy, feeling fear, trying to reach a safe place to hide (and increase safety need). The second one will feel angry and will try to defend the house positioning in front of the enemy until this has gone (satisfying need 2).

In this case we introduced the strong increase in the intensity of the emotion. So even in the case he feels very happy because of its owner, the emotion will turn into much fear or anger (depending on the personality).

Experiment results:

When AIBO processes the strange sound, effectively the need 2 experiments a high decrease. So this need becomes the most priority and the scripts executed will try to satisfy this need. In both personalities, the AIBO goes to the door, sniffing the strange smell. Then, he finds the smell came from the enemy, so in that moment is when the big increase of the emotion takes place. This means that even if the mood is positive, the emotion of fear or angry will be clearly shown.

AIBO behaves as expected: the friendly flees with high fear values and the unfriendly stands very angry facing the enemy. These situations make both AIBOs to increase their values of needs 2. When the influence time of the event finishes, both AIBO can change the emotion to one positive (depending on the most priority need to satisfy). In both cases, since the mood was very negative, the positive emotions are not shown. Some time must go by, until the mood starts to be positive and the emotion has any value.

In the case AIBO was sad before the 'strange sound' because he could not find its owner in Scenario 3, after the enemy threat, he will come back to that state of waiting bored to the owner. Since sad is a negative emotion, the decrease of the mood value will make AIBO show stronger sadness.

6.6 Survey evaluation and results

This section exposes the different results obtained in the survey done by 24 people when watching four different videos. In those, AIBO performances in the four situations explained in previous section. The survey treats to give light over some subjective measurements that complete the evaluation of the prototype. This section also analyzes the results for further improvements.

This section tries to evaluate some subjective aspects included in our requirements (Table 20). The experiment has been done following the scenarios of the previous section. Four different videos have been shown, where each one contained the recording of one of the scenarios. The third first videos were shown as continuous chapters of the same story. The fourth one was recorded independently one for a friendly AIBO and another for an unfriendly AIBO.

The people who participated in the experiment had to answer some questions after watching a 10 minutes video. These questions have been chosen following the aims stated in previous section to give some clues about the accomplishment of the requirements. The document that the people had to fill can be found in Appendix I (9.8) related to the aims we wanted to evaluate.

In the next section we included the graphics of the answers of the people to analyze briefly the results. Even if the questionnaire was no conclusive, it has helped to get a better understanding of what a companion dog must provide to the user.

6.6.1 Events response

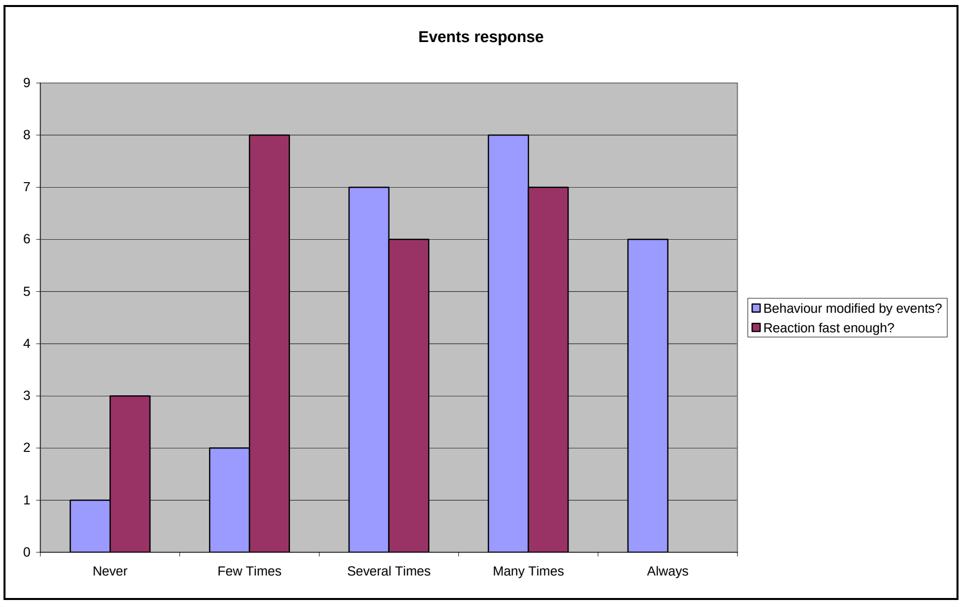


Figure 63: Events response graphic

According to Figure 63, we can see that even the people think the events really modified the behaviour, but the speed of the response is not enough. We talked before (section 6.3) that the response can be speeded up only when reducing the number of actions per script. This will decrease the time between two reasoning iterations. In that case the only problem will be the constraints of the speed in robot movements, that cannot be overcome until a new faster platform is available.

Another solution can be to convert the reasoning process in an independent thread. This means that the CCU will not control the iterations of the reasoning directly. This is a good solution for a commercial product, but for researching process, the author discourages this solution because makes more complex the overall timing control of the system, so strange behaviour may appear.

6.6.2 Companion experience

The results show that some incoherences were found in AIBO's behaviour when acting in the different scenarios. This is not casual due to: first, the speed of the robot, second, the fact that actually is a robot an it has not tongue, neither eyes, …and third, that the author is not an expert in dog ethology. Although we researched in that field there is not a defined behaviour for each situation. Besides the people who evaluate the robot thought in their own experiences with dogs, and every dog is different, so they gave some comments that sometimes were even contraries (for example, reaction in front of the food). Despite some incoherences were found, the global results for AIBO as a companion dog (blue bars) show that was convincing the most of the time, so we accomplished our main goal successfully.

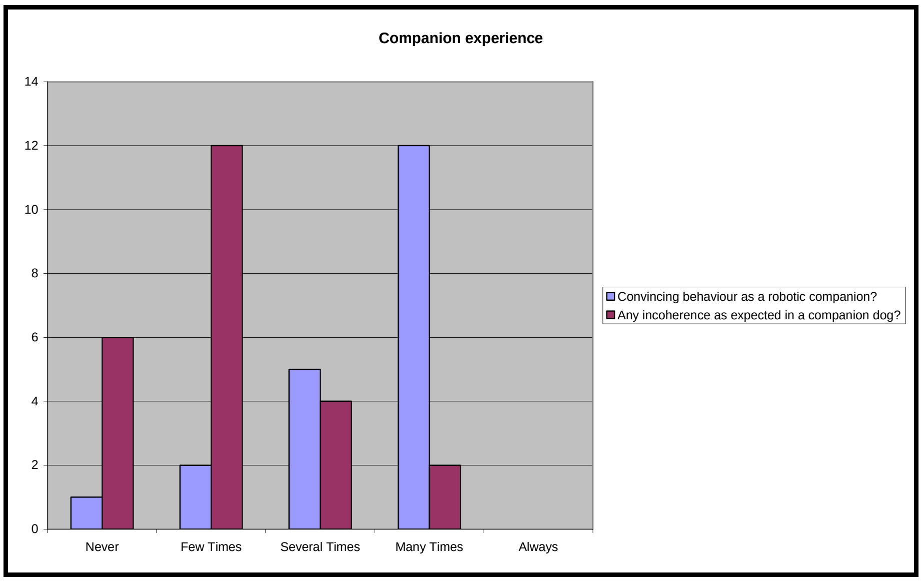


Figure 64: Companion experience graphic

6.6.3 Children adapted

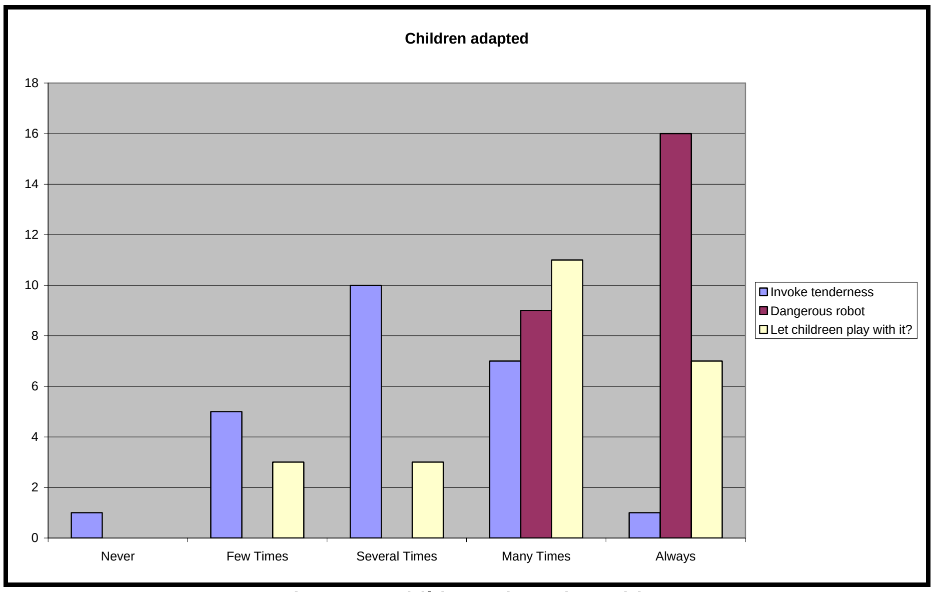


Figure 65: Children adapted graphic

Although this question depends a lot on the AIBO's behaviour and in the scenarios he showed anger only once, there is an important issue in this point. This is that the aspect of the AIBO, the design made by Sony, hardly can produce fear in an adult. This has to be evaluated also using children to see their opinions. Even with that we can say that our objective was also achieved.

6.6.4 Personality distinguished

We showed two videos, with four chapters each one. Only the reactions of the dog were different in the last chapter. Eleven people watched the first video (friendly) and thirteen people watched the second one (unfriendly).

The first one was well classified by 7/11 and wrong by 2/11. In the second one 6/13 classified correctly the personality of the dog, while 7/13 did it wrong. This can be explained because the video only showed some anger in AIBO when facing an enemy, and the situations were not so complex. However, this can be also a sign of the difficulty of creating a complex personality using only the data gathered by the author. Bartneck must be mentioned again and its idea of including a member of the team, totally focus in the development of personality.

We do not think that the approach of the 'fixed' personalities is wrong. The problem is how to create sets of actions that are appropriate for a particular type of person and no for another one. A wide research should be done in this psychological field to set explicitly the features of each personality. In addition many examples in real situations must be provided to help in the process of facing new situations.

6.6.5 Emotional responses

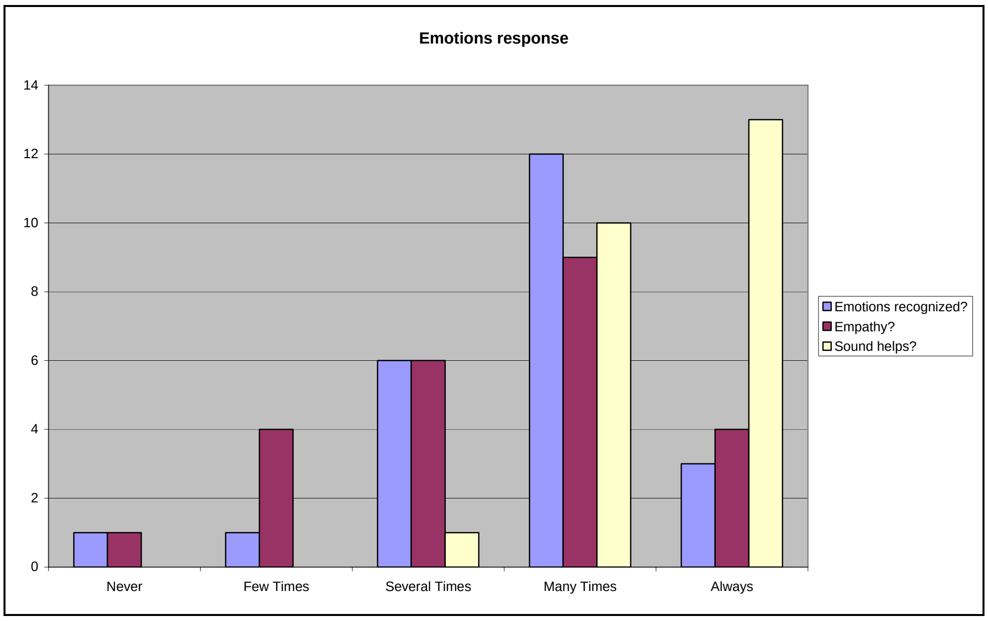


Figure 66: Emotions response graphic

According to the results shown in Figure 66, the creation of the levels of emotions has been successfully accomplished, and according to the answers to question 13, the very most of the people agreed in the fact that good sounds helped a lot. So real sounds are essential to improve the emotional recognition process.

6.6.6 Situation awareness

The graphic in Figure 67 shows that although the adaptation of AIBO to the situations we proposed was not perfect during all the demonstration, the most of the time showed an appropriate behaviour according to the circumstances. Besides only few times showed strange reactions, so a coherent performance of AIBO when evaluating each situation was successfully accomplished.

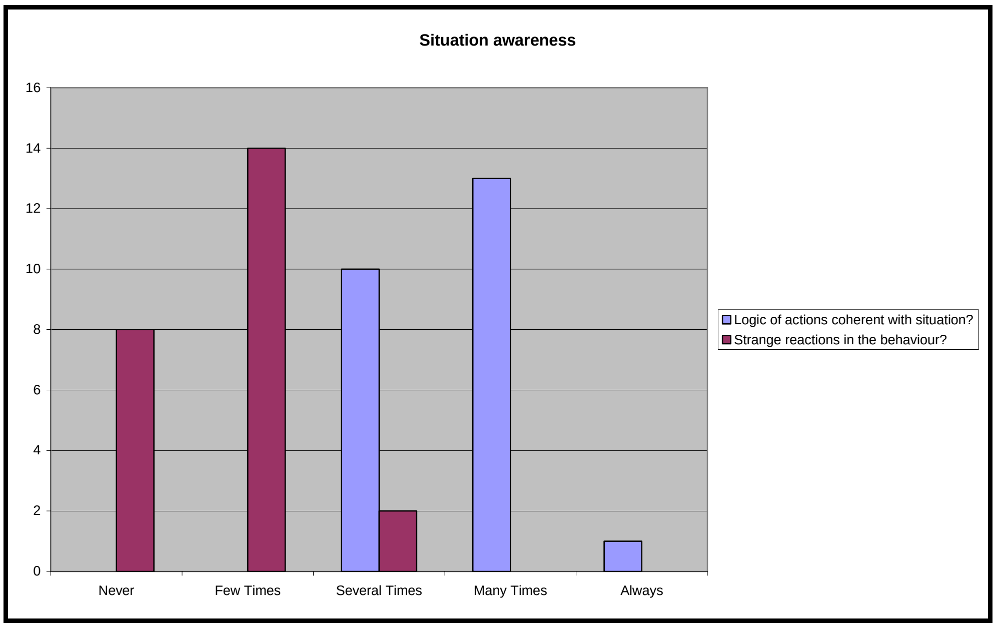


Figure 67: Situation awareness graphic

7 Conclusions & Recommendations

As the result of human curiosity, the future will bring us a deeper knowledge about human mind, and combining with the advance of technology, this knowledge will be materialized in the creation of new social robots. In this thesis we have create a model of personality based on the satisfaction of internal needs. This model has been tested using a prototype thanks to the creation of a new framework that will allow further research.

At this point, when the model was exposed and some experiments over the prototype were carried out, is necessary to recapitulate the achievements of this thesis together with its drawbacks to set some conclusions that aid the development of future models of artificial personalities. This chapter provides in section one the conclusions extracted of all the work already done. In section two, the author gives a set of recommendations that can improve the current system or give some clues for the development of new ones.

Chapter overview:

7.1 Conclusions

This section gives a very brief summary about how the system works and exposes the pertinent conclusions according to the different stages that have been followed to carry out this project: analysis, design, implementation and evaluation. The very general conclusion is that the most of the requirements have been successfully accomplished; however some choices such as the use of expert systems even if we obtained good results, can fail when facing a more complex system.

Needs based companion AIBO is a research in developing a personality model for a companion robotic dog. It also includes the creation of a flexible framework that allowed the implementation of that model in a real prototype. This prototype was used to test the theoretical model in a commercial robotic platform called AIBO.

The developed companion AIBO acts as an artificial agent that is able to gather information about its environment (real world) with a GUI that can be managed by the researcher. AIBO will use this information, combining with its internal mental state to infer an appropriate set of actions to perform coherently in the real world. Its behaviour is driven by the satisfaction of three internal needs: *physiological, safety, love & belonging*, as a part or the *nPME* model. Combining the needs with the PME components, a personality is created in AIBO companion dog. Thanks to this personality, in similar situations two AIBOs with different personalities will act differently. In addition, the personality model allows the robotic dog to express different emotions with appropriate intensities, what increases the realism of the companion experience. In this process of creating an artificial personality for a companion AIBO, an architecture of a system that allowed us to create a prototype was developed. The architecture focuses on the isolation of its different modules with the objective of making a flexible framework that can be used not only for the nPME model, but also for new mental models that can be developed in the future. This is our best contribution for future researchers.

The final companion AIBO, as a result of the implementation of the nPME model inside the reasoning module of the framework, has been tested in different ways. After a comparison with the Sony commercial AIBO mind software, the prototype was also tested using four defined scenarios. Its behaviour was recorded and a doing an extensive user study with 24 different people (mainly students) we found out if our companion robotic AIBO behaves similarly to a real companion dog. Thanks to the survey we can provide some useful ideas about how to improve the companion experience.

The experimental results have shown that our approach has been correct in many points, but they have also shown that there is much work to do for creating a real robotic companion dog. Let's talk more in detail about the most important stages in the developing of this thesis.

7.1.1 Analysis stage

At the beginning we start gathering some information about current research in domestic robots. Through the lecture of several papers we could figure out what was the concept behind a companion robot. That allowed us to establish the first set of requirements that delimited the problem to solve.

When the problem was stated, in chapter 2 we gave a complete summary of the main features of the AIBO robot, explaining its many advantages as well as its drawbacks. We studied the approach that Sony made to create the commercial AIBO's behaviour. Much information about *Personality Model for a Companion AIBO* [3] was included since the initial nPME personality model exposed there, was the basis of the current work. Then we also showed different approaches of the mental model that were motivated to overcome some difficulties when implementing the original nPME model. In addition, we provided the necessary technical background to make the reader familiarize with concepts as knowledge engineering, artificial agent, BDI model, etc. This was so important because it provides the essential data to consider when deciding how to design the system and which tools are most suitable for implementation. We cannot forget that every theoretical model must be tested in a real prototype to check its validity.

In chapter 3 we analyzed all the information provided to build a robotic companion dog. That information was classified in three different areas of knowledge. The *technical analysis* gives light to the decisions we took when choosing the different tools for implementing our system, always trying to keep a balance between the lowest prices of the tools to used and the power of such tools (Java, URBI programming languages). In addition we chose the AI technique of expert systems for the developing of the reasoning system over the basis of the BDI model. The *use and interaction* analysis pointed out some requirements that a realistic companion robot must achieved, such as continuous behaviour, autonomy, etc. Last section analyzed the *psychological* models that were explained in Chapter 2, where the advantages and disadvantages of each one were highlighted. The goal of this analysis was to extract the best ideas from each model to combine them in only one approach. We saw that the nPME model was a promising model to develop but some modifications were also necessary (variable reduction, amygdale approach) to use all its potential.

7.1.2 Architecture of the system

This stage in the developing process was crucial not only for our prototype, but also to allow other researches to continue our work providing them a useful framework where the implementation of their own mental models was possible. To achieve this, a modularized architecture was designed. It was composed by five different modules: *events m., scripts m., reasoning m., connection m.,* and the *Central Control Unit* module.

Among these components the most important modules for our purposes were the *Reasoning module* and the *Central Control Unit* or *CCU*. The reasoning module will be commented in next paragraph (7.1.3). The second one, called the CCU, was a crucial element in our framework. It worked as an intermediary to exchange data between different modules. Thanks to such a centralized approach, the isolation of each module could be improved. The consequence is that a new module can be integrated in the system in a quite easy way. In addition the CCU manages the main timing line in the system. We have also seen that until certain extent this can be less efficient in terms of the overall speed, but at the same time provides a very easy and simple way of controlling all the system, and it allows modifications in just one file when the timing has to be changed. For a commercial application independent threads could be advisable, but from the researcher point of view this feature gives an important and useful characteristic: easy control. Every modification in the system will be also easily integrated thanks to this component.

The external events are translated into an understandable language for the system thanks to the combination of the *Connection module, CCU* and *Events module*. This translation was one of the premises when starting with the project since we wanted an autonomous agent performing in the real world. We can say that this translation is partially successful. Instead of real events, we use a GUI that provokes them, so that is a shortcoming. However the GUI was also built in the C*onnection module* in a way that simulates the information that can be gathered from real stimulus. So the information that constitutes the output in the *Connection module* is exactly the same than in the case of real stimuli. With those data, the CCU is able to create new events in the *Events module* and also introduce/remove them when necessary into/from the reasoning system.

7.1.3 Reasoning system

The reasoning process has been implemented in the R*easoning Module*. This is where the theoretical nPME model is materialized in a computerized process. To implement the model, two highly coupled inference engines were used as expert systems. In this implementation the needs are the basic element that leads the performance of the system.

The first inference engine chooses according to the previous values of the needs (modelled as buckets that must be filled to satisfy a certain need), the most priority one. For the needs creation we have based on *Maslow's pyramid of needs*, where a need can be satisfied only if previous ones (more priority) have been already satisfied. The first engine also includes the *amygdale approach*. This is a new idea that has been developed to prioritize some events in a simple way that may have fatal consequences in the life of our artificial being. For example, when an enemy enters home, AIBO should focus on fleeing or fighting (these will depend on its personality), even if he is satisfying other need. This approach obtains faster results when facing certain events. In the implementation the amygdale is seen as the first layer that can reduce (or sometimes increase) quickly the value of some needs when specific events occur. It works before the system chooses the need to satisfy, so the amygdale influences directly that choice, and prioritize some decisions.

The second inference engine is selected among a group of personalities. These are created using some descriptive information about the behaviour of each personality. This relies on the assumption than certain personality will behave always in a certain line of actuation. This line can have peaks and valleys that are modelled using the *mood*. This is one dimensional axis that will vary according to the situation, setting the intensity of the different *emotions* that the robot can show. Therefore, different personalities in the same situation will act differently and the events will increase or decrease the mood and emotions also in different amounts. Besides, the type of emotions shown when facing certain stimulus can be different. It also happens when the degree of satisfaction of every need, that can give different. So, using this approach we assure that the behaviour of other personality under same circumstances can be slightly or totally different.

Another Graphical User Interface was also built in this module to show all the variable evolution during time when the robot is performing in the real world. Thanks to that we can follow the different level of the needs, the value of mood, the emotion that is showing in each moment together with its intensity. Besides, the script that the reasoning system has chosen to execute is also shown with a little description to favour the understanding of the theoretical model behind the implementation.

7.1.4 Prototype

The prototype was implemented using Java as the main component for the control of the system, Jess for the reasoning module, and URBI to control the physical behaviour of the AIBO robot. A big effort was made to complete the actions taking by the reasoning system with appropriate scripts that shows a realistic behaviour. More than 150 new functions have been created to show different movements in combination with sounds for AIBO. Every script is a combination of those functions together with one emotional expression of a different intensity. The new developed actions have revealed the importance of realistic sounds to identify AIBO actions or to empathize with AIBO emotional state, providing a very richer companion experience.

AIBO is able to choose a state in a set of finite state diagrams according to its personality, the level of the needs, the events and the value of the mood. The most important choice for a certain personality is the need to satisfy, and then its behaviour is further detailed thanks to the events present on memory. Using that the reasoning system will choose an emotion up to 6 to show (with five different intensities). The intensity is shown when combining different actuators: face leds, mouth, head, neck, tail and sounds. Then the emotion acts during the script execution, that comprises a set of ordered actions such as sit, walk, play, etc.

Since the prototype can not perceive real events in the environment, the prototype uses a GUI where every event can be trigger just by pushing a button. However, the system allows the future users to implement the real integration of the events juts by modifying the module that controls AIBO. The inclusion of the event into the system will be done automatically by the rest of the modules and no further modifications must be done.

7.1.5 Tests

Different test were carried out to test to check the validity of our needs-based approach, the correctness of the implementation and the performance of the prototype.

First part in those experiments showed that AIBO performances logically as expected according to the criteria applied in the code. The actions of the companion AIBO followed a coherent path according to the personality, the level of the needs, the events and a certain mood. If a new event occurs during AIBO performance AIBO is able to change its action plan and performance new actions that lead at the end in an increase of its needs. The different tests applied to the system have shown that the modification of the scripts is so easy thanks to the fact they are wrapped into simple text commands, and the translation into URBI is made automatically.

The second evaluation was made using Sony's AIBO Mind Software in comparison with our system. Despite the fact the model of Sony gives the chance of real interaction through robot sensors, the results show that the emotions are hard to recognize, using electronic sounds, and there is no an on-going emotional state neither a set of well-defined actions when facing events. In comparison our approach give six well defined personalities with different levels that can be identified quite easily thanks to realistic sounds.

The third part of the evaluation comprised four particular scenarios where the behaviour is carefully described. The behaviour of AIBO when facing the different situations (food, owner, enemy) works as expected with the precise change of needs when a new event provoked it and the emotion expression for every script. Mood also evolves according to the situation and influences the intensity of the expresses emotion. These scenarios were recorded in a video.

The last part of the evaluation consisted in presenting the video to two groups of people. Each video showed a different personality. The attendants filled after the video a survey questionnaire about their impressions when watching a robotic dog acting as a companion.

So, after the experiments we can conclude that the architecture of the system was a successful approach. The system performs correctly and coherently in every scenario. The CCU acts pretty well as an intermediary and it has been revealed as a useful tool when some modification in the system had to be introduced. The scripts and basic functions defined in simple text files have been a very successful approach for the testing stage. It is possible to change the parameters and actions of every function and script in a very easy and fast way. So thanks to this speed in modification the behaviour of AIBO has been really improved from the first stages.

The use of expert systems for the reasoning module has given a successful result in both faces: first, the simplicity when a rule had to be changed, or one parameter had a wrong value, and second, the power and speed in the reasoning process that is so necessary for a system that works on real time. However the complexity of the mental model had to be reduced to let its implementation in the expert system. Although the rules can be easily written, there were many variables to take into account during the reasoning process, what makes it sometimes difficult to adjust. Besides, in this prototype the amount of situations AIBO could encounter was relatively small. When this number increases the task of writing every rule can become a very difficult process, despite the good results we achieved.

The results of the user study through the survey show that in spite of few incoherences that were found in the behaviour, the most of the time, AIBO was acting successfully as a companion dog, showing different responses in the presence of new events as well as an emotional behaviour according to the circumstances. The analysis of the personality suggested that although AIBO showed different behaviours in both videos, the personality was hardly recognized. The author proposes a careful research in the behaviour shown by several real dogs with different personalities before facing the modification or the implementation of new ones. AIBO also has shown that according to its physical design (as a toy), it hardly can be seen as a threat by adults, so aggressive behaviours will be harder to show. The results showed also that the coherence in the behaviour was also successfully accomplished getting the right balance between the robotic limitations and a real companion pet.

7.2 Recommendations for future work

This section advices some useful recommendations for further work. The recommendations are suggested from both points of view: technical improvements for the whole system, and new ideas for a newer psychological model.

After all this time of work, many improvements have become clear. Some related to the technical part, and some to the psychological, that at the end they join all together to get the whole improvement.

The first improvement it is to increase the set of primitive actions with more subtle movements what will surely enhance the overall performance of the system.

Second improvement, it is of course trying to embed the whole system into AIBO robot to make it totally autonomous. The system we have implemented would need some modifications as the inclusion of an *autonomous navigation system* and the translation of the external events into the events code that the system understand. This will improve highly the interaction, what is essential for a companion dog.

The reasoning module gave good results but it would be great to make some modifications on it. First, try to keep all the emotions values stored during execution. Each state will modify every emotion in a certain amount, and then a function, in every state, must decide which emotion should be shown. The idea of blending emotions is also advisable but the constraints of the AIBO robot will make the blended expressions very difficult to combine.

A learning module can give many advantages to the system and it is actually a possibility with the current model. First thing to do would be the definition and implementation of many different lines of action when AIBO is facing a situation. AIBO could choose at the beginning each one in a random way. If thanks to the choice its needs and mood increase their values, the learning system would weight heavily the probability of chosen that path for the next time.

It is also highly advisable to make a bigger separation in the reasoning system between the personality-emotional model and the situation election model. An architecture with three modules would be useful. One of them must get the emotion and the changes in the needs according to the events, the other one set a sequences of actions or plan to achieve the satisfaction of the chosen need but without the emotional response. The third one should act as an intermediate merging both responses and combine them with a navigation system to provide a full-autonomous response.

We propose the creation of an independent amygdale module. The author really encourages the idea of the amygdale, because it has been so important in every complex creature to ensure its surveillance. In this case it will be the first module to process information when an event has occurred. This would consist in a few simple but drastic scripts that would make AIBO perform some basic actions without another kind of reasoning. We can match this idea with very intensive emotions. If AIBO would need a fast response, *amygdale module* would give it to him. In normal situations, after this amygdale filter, the normal reasoning would take part, cushioning the first impulses when they were not so strong.

Another idea for the reasoning module can be based on a *Freud approach*: This improvement is a bit similar to the amygdale approach. In this case, there have to be three different reasoning modules. One responsible to instincts and pleasure what is called '*Id'* in Freud's terminology. Another one with all the social rules and learnt behaviours that can be called '*SuperEgo*'. The third one would be '*Ego'* that basically will weight the decisions taking in every situation by the other two engines to choose only one according to the external circumstances.

Another improvement that can be useful would be to extend the dog needs using the fourth and fifth need from Maslow. This has not so much sense in a dog, but although it may not be conceptually right, it will bring to the robot behaviour a wider and richer behaviour. The only thing to do using these would be to identify the new needs (esteem and auto-realization needs) with some realistic needs of the AIBO. Esteem can be match with the gregarious behaviour of a dog, giving him more resources to analyze the situation when encountering more dogs. The auto-realization module can be associated to the moments he wants to play, and the rest of the needs are satisfied. Besides this, the third need can be completed with the sexual instinct to give more realism to the behaviour of AIBO.

The author also recommends the use of other AI techniques alone or in combination with the expert systems to provide more realism to the behaviour. One immediate example can be the use of fuzzy logic techniques to weight more accurately the intensity of mood and emotions. The principle of genetic algorithms can be used to create a more complex set of behaviours from the random combination of primitive actions. In combination with the learning module can give theoretically great results.

8 References

[1] J. McCarthy, *"What is Artificial Intelligence"*. Revised November 24, 2004. http://www-formal.stanford.edu/jmc/whatisai/whatisai.html

[2] M. Wooldridge, "*Reasoning About Rational Agents*". The MIT Press. 2000.

[3] I. Dobai, L. Rothkrantz, C. van der Mast, "*Personality model for a companion AIBO*", *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ISBN 1-59593-110-4, pp. 438-441, ACM, Broadway New York, June 2005.

[4] *Executive Summary*. World Robotics 2006. - http://www.euron.org/miscdocs/stats06.pdf

[5] "*Trust me,* I'm a robot", The Economist print edition. Jun 8th 2006.

[6] Sony AIBO Global web site - http://www.sony.net/Products/aibo/

[7] M. Fujita, "*On Activating Human Communication With Pet-Type Robot AIBO*", Proceedings of the IEEE,Vol.92, NO.11, November 2004

[8] M. Fujita, "*AIBO: Towards the Era of Digital Creatures*", International Journal of Robotics Research, 20(10):781-794, October 2001.

[9] M. Heerink, B. Kröse, V. Evers, B. Wielinga, "*The Influence of a Robot's Social Abilities on Acceptance by Elderly Users".*Amsterdam, 2006

[10] Z. Yang, B.Tsing, L.J.M. Rothkrantz, "*AIBO as a watchdog*", *Game-On 2006*, vol. Game-On, no. 7th, pp. 74-78, Eurosis, December 2006.

[11] D. Datcu, M. Richert, T. Roberti, W. de Vries, L. Rothkrantz, "*AIBO Robot as a soccer and rescue game player*", *Proceedings of GAME-ON 2004*, ISBN 90-77381-15-5, pp. 45-49, November 2004.

[12] I. Borm, L. Rothkrantz, "*A multi-agent soccer simulatior based on a simplified soccer model*", *Game-ON*, ISBN 9077381317, vol. 7th, pp. 49-54, Eurosis, December 2006.

[13] URBI official web page - http://www.urbiforge.com/eng/index.html.

[14] K. Dautenhahn, S.Woods, C.Kaouri, M.L.Walters, K.L.Koay, I.Werry, "*What is a Robot Companion – Friend, Assistant or Butler?,* Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE

[15] B. Friedman, P.H.Kahn, Jr, J.Hagman, "*Hardware Companions? – What online AIBO Discusión Forums Reveal about the Human-Robotic Relationship*". 2003, CHI 2003. Volume Nº5, issue nº 1.

[16] A. Pozuelos, "*Criando a mi perro (Breeding my dog)".* Punta veterinarian centre. 2006.

[17] Sony AIBO Global web site - http://www.sony.net/Products/aibo/

[18] Sony AIBO European web site - http://www.eu.aibo.com/

[19] A. Damasio. "*Descarte's Error: Emotion, Reason and the Human Brain*". G.P. Putnam's Sons, 1994

[20] R.C. Arkin, M. Fujita, T. Takagi, R. Hasegawa. "*An Ethological and Emotional Basis for Human Robot Interaction*" Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. 2003.

[21] P. Eckman, R.J. Davidson, "*The Nature of Emotion*", Oxford University Press, 1994.

[22] A. Takanishi, "*An Anthropromorphic Robot Head having Autonomous Facial Expression Function for Natural Communication with Human*", 9th International Symposium of Robotics Research (ISRR99), 1999, pp. 197-304

[23] Sony AIBO development tools - https://openr.aibo.com/openr/eng/perm/main_menu.php4

[24] R.L. Beaumont, "*Five Factor Constellations and Popular Personality Types*", Psychology 106, 2003

[25] C. Bartneck, "*Integrating the OCC Model of Emotions in Embodied Characters*", *Workshop on Virtual Conversational Characters*, 2002

[26] P.J. Lang, "*The Emotion Probe: studies of motivation and attention*", A study in the Neuroscience of Love and Hate. Hillside, NJ: Lawrence Erlbaum Associates, Publishers, 1995.

[27] A. H. Maslow, "*Motivation and Personality*", 2nd. ed., New York, Harper & Row, 1970

[28] M.P. Georgeff, F.F. Ingrand. ***"**Decision-Making in an Embedded Reasoning System**",*** Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit (Michigan). 1989.

[29] Programming languages generations: http://searchsmb.techtarget.com/sDefinition/0,,sid44_gci211502,00.html

[30] Computer user – High Tech dictionary - http://www.computeruser.com/resources/dictionary/

[31] Expert Systems - http://www.pcai.com/web/ai_info/expert_systems.html

[32] www.wikipedia.com

[33] Web page that gives information to add new behaviour to AIBO robots. They also provide some useful info about Sony's Behaviour Model. - http://www.aibohack.com/

[34] C.H.A. Lee, "***Emotional AIBO***", Internal Report, 2004.

[35] S. J. Russell, P. Norvig. "***Artificial Intelligence: a modern approach***", 2nd Edition. Prentice Hall, 2002.

[36] http://searchwebservices.techtarget.com/

[37] UML group - http://www.uml.org/

[38] Information about useCaseDiagrams - http://www.agilemodeling.com/artifacts/useCaseDiagram.htm

[39] Real time systems information - http://www.dedicated-systems.com/encyc/techno/terms/defini/def.htm#realtime

[40] More documentation about URBI : http://www.gostai.com/urbi.html

[41] JESS, the rule engine for Java Platform - http://herzberg.ca.sandia.gov/

[42] CLIPS, a tool for building Expert Systems - http://www.ghg.net/clips/CLIPS.html

[43] Friedman-Hill, E., "Jess in Action. Rule-Based Systems in Java", Manning Publications Co., 2003

[44] Sony's documentation is no longer available. So for the user will be necessary to look in no-official pages.

[45] Tekkotsu framework. http://www.cs.cmu.edu/~tekkotsu/

[46] B. Eckel., "Thinking in Java", Prentice-Hall Inc, 2000, 2nd edition.

[47] P.Y. Oudeyer, F. Kaplan, V.V. Hafner, A. Whyte, "*The playground experiment: Task-Independent Development of Curious Robot*", In the Proceedings of the AAAI Spring Symposium on Developmental Robotics, 2005

[48] G. Boeree, "*PersonalityTheories*", 2006. http://www.ship.edu/%7Ecgboeree/perscontents.html

[49] Brief Summary of Personality Models
http://changingminds.org/explanations/personality/personality.htm

9 Appendixes

9.1 Appendix A: Glossary

In this appendix we provide some definitions about some of the more important concepts that have appeared in the report. The goal of this section is to provide easy, basic definitions for readers that do not have a high level of technical knowledge.

1. ***Expert System***: also known as a knowledge based system, is a computer program that contains some of the subject-specific knowledge of one or more human experts. (Source: www.wikipedia.org)

2. ***Inference Engine:*** is a computer program that tries to derive answers from a knowledge base. It is the "brain" that expert systems use to reason about the information in the knowledge base, for the ultimate purpose of formulating new conclusions. (Source: www.wikipedia.org)

3. ***Unified Modelling Language:*** is the model language more used currently for software systems (although the fact is not an official standard). It is a graphical language to visualize, specify, build and document a software system. (Source: www.wikipedia.org)

4. ***Agent:*** It is an "encapsulated computer system, situated in some environment, and capable of flexible autonomous action in that environment in order to design objectives" (Wooldridge).[2]

5. ***Artificial Intelligence: "***the science and engineering of making intelligence machines". (John McCarthy)

6. ***Event:*** it is a software message that indicates that something has happened. It can be an external event (it occurs out of our system) or internal one (caused by the own system). (Source: www.wikipedia.org)

7. ***Java:*** "is a programming language developed by Sun Microsystems. Java applications are typically compiled to bytecode, although compilation to native machine code is also possible. At runtime, bytecode is usually either interpreted or compiled to native code for execution, although direct hardware execution of bytecode by a Java processor is also possible." (Source: www.wikipedia.org)

8. ***Jess:*** it is a rule engine for Java platform, based in CLIPS that allows to build an expert system using all the features that Java provides. It is written entirely in Sun's Java language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA [41].

9. ***URBI:*** (Universal Robot Body Interface) "**URBI** (Universal Real-time Behavior Interface) is universal robotics platform based on a new scripted language, designed to work over a client/server architecture in order to control a robot or, in a broader definition, any kind of complex system, like a video game environment or characters." [13]

10. ***AIBO:*** Artificial Intelligent RoBOt. Developed by Sony, it is an autonomous robotic dog. It has some sensors to recover information for its environment that allows it to avoid obstacles or recognize some images and sounds. It has also actuators to interact with the real world. It has a computer mind that can be programmed by the user.

11. ***System Architecture*:** "The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution." From ANSI/IEEE 1471-2000.

12. ***Knowledge based system*:** "It is a computer system that is programmed to imitate human problem-solving by means of artificial intelligence and reference to a database of knowledge on a particular subject. Knowledge-based systems are systems based on the methods and techniques of Artificial Intelligence. Their core components are the knowledge base and the inference mechanisms." (Source: www.wikipedia.org)

13. ***Cognition*:** this concept references the capability of beings to process information from perception, the acquired knowledge and some subjective features that let to evaluate some aspects instead of other ones. (Source: www.wikipedia.org)

14. ***Cognition*** (in psychology and artificial intelligence)**:** functions, processes and mental states of intelligent agents, more focused on process such as understanding, inference, decision making, planning and learning. (Source: www.wikipedia.org)

15. ***Metacognition*:** It refers to thinking about cognition or to think or reason about one's own thinking. (Source: www.wikipedia.org)

(Note: many of these definitions are extracted from WIKIPEDIA. We have chosen this source because it gives an easy way to define concepts for people without much technical background. Although this is not an official encyclopedia, the contents we exposed here have been contrasted with other sources and also with the knowledge of the author along his years of university experience.)

9.2 Appendix B: Scenario Diagrams

I. Friendly personality

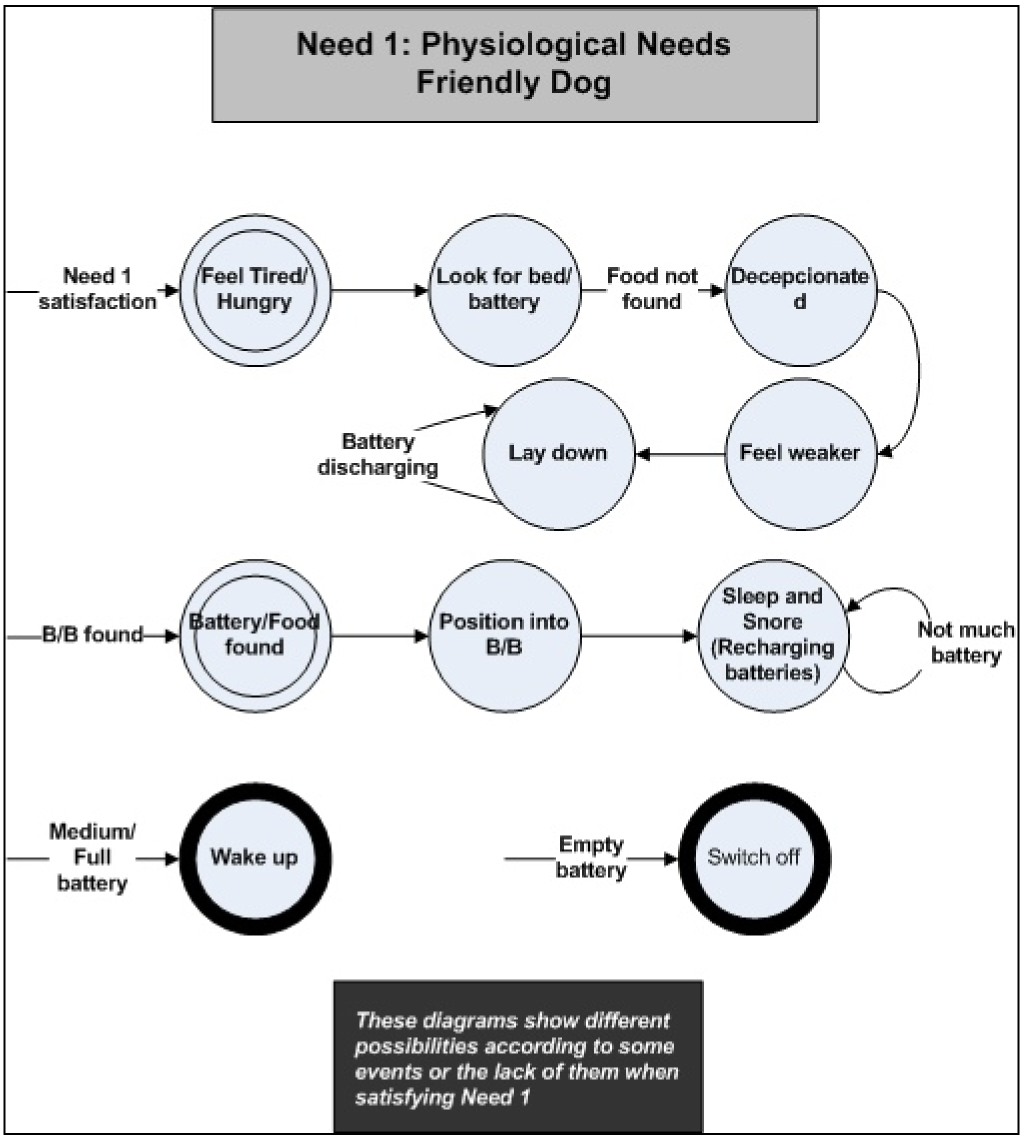


Figure 68: Need 1 – Friendly

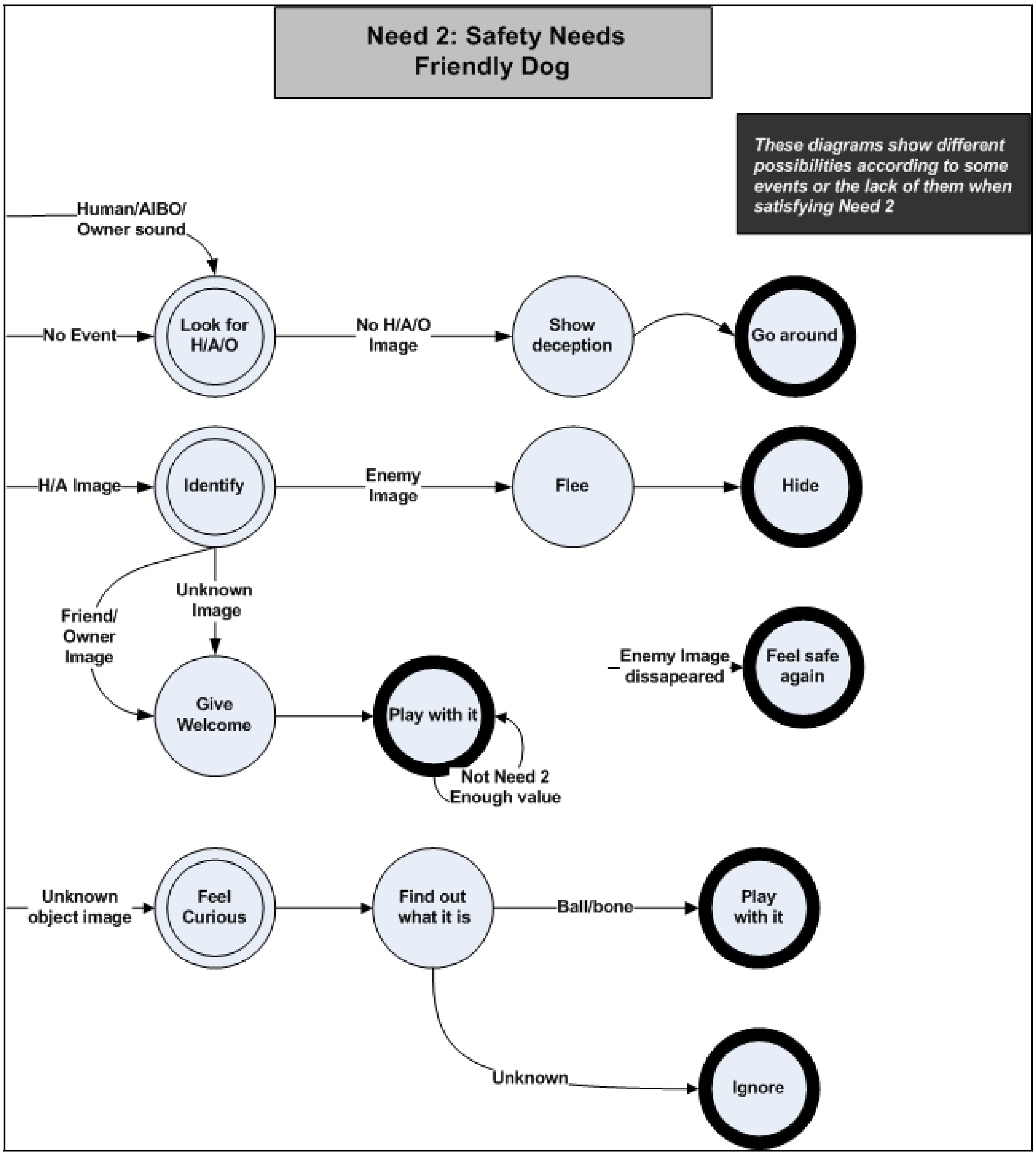


Figure 69: Need 2 – Friendly

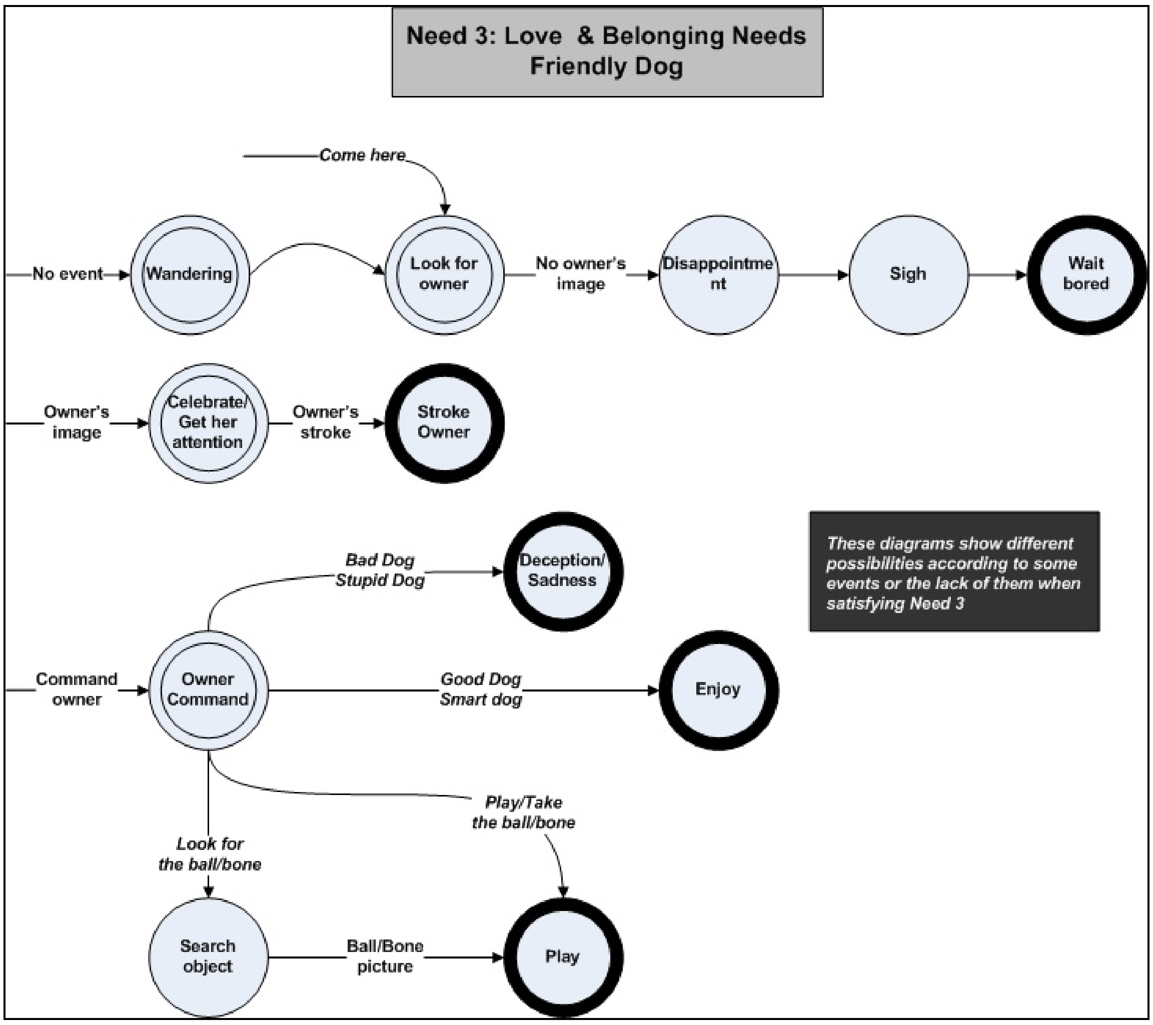


Figure 70: Need 3 -Friendly

II. Unfriendly personality

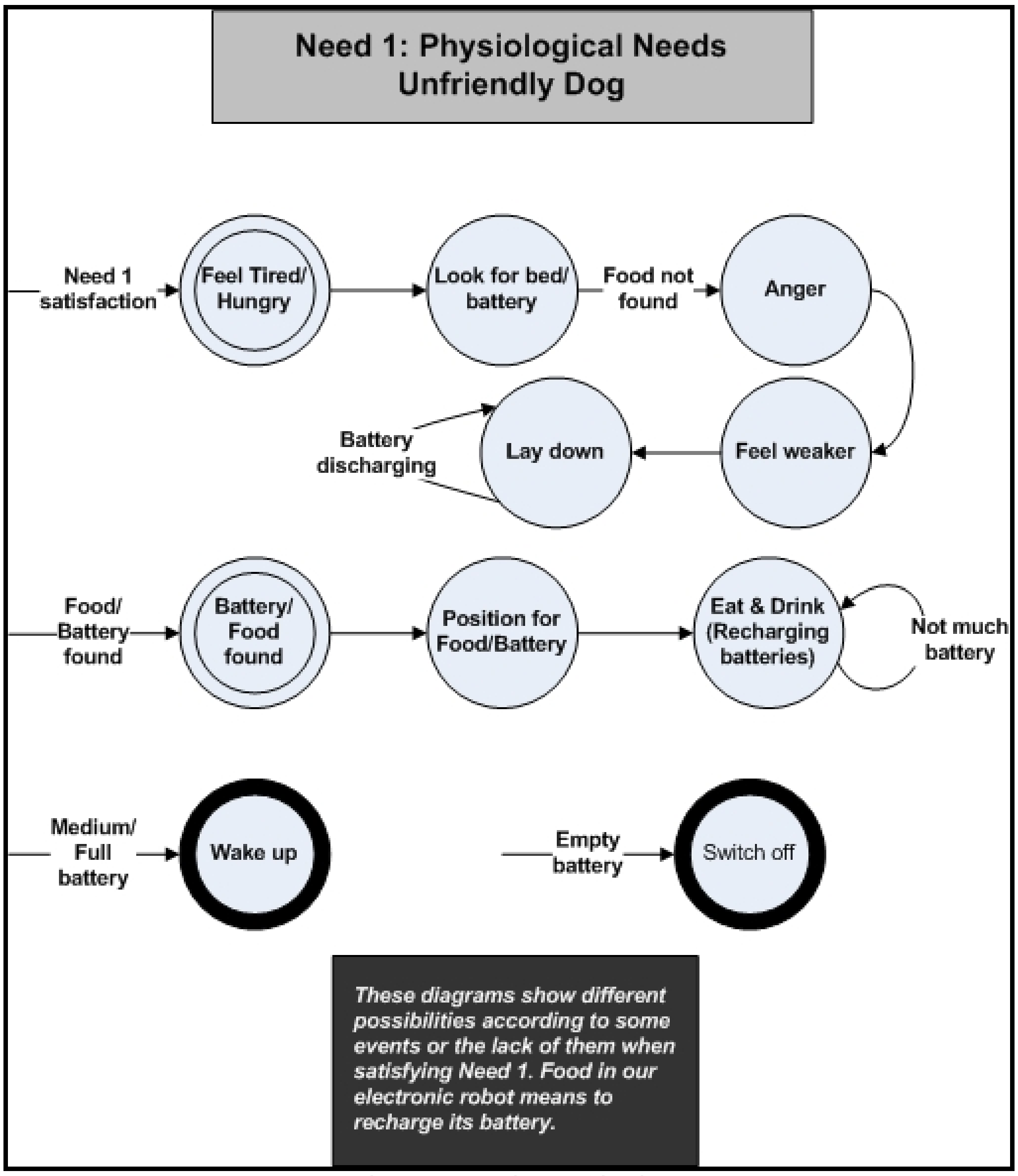

Need 1: Physiological Needs
Unfriendly Dog
Need 1 satisfaction
Feel Tired/ Hungry
Look for bed/ battery
Food not found
Anger
Feel weaker
Lay down
Battery discharging
Food/ Battery found
Battery/ Food found
Position for Food/Battery
Eat & Drink (Recharging batteries)
Not much battery
Medium/ Full battery
Wake up
Empty battery
Switch off
These diagrams show different possibilities according to some events or the lack of them when satisfying Need 1. Food in our electronic robot means to recharge its battery.

Figure 71: Need 1 - Unfriendly

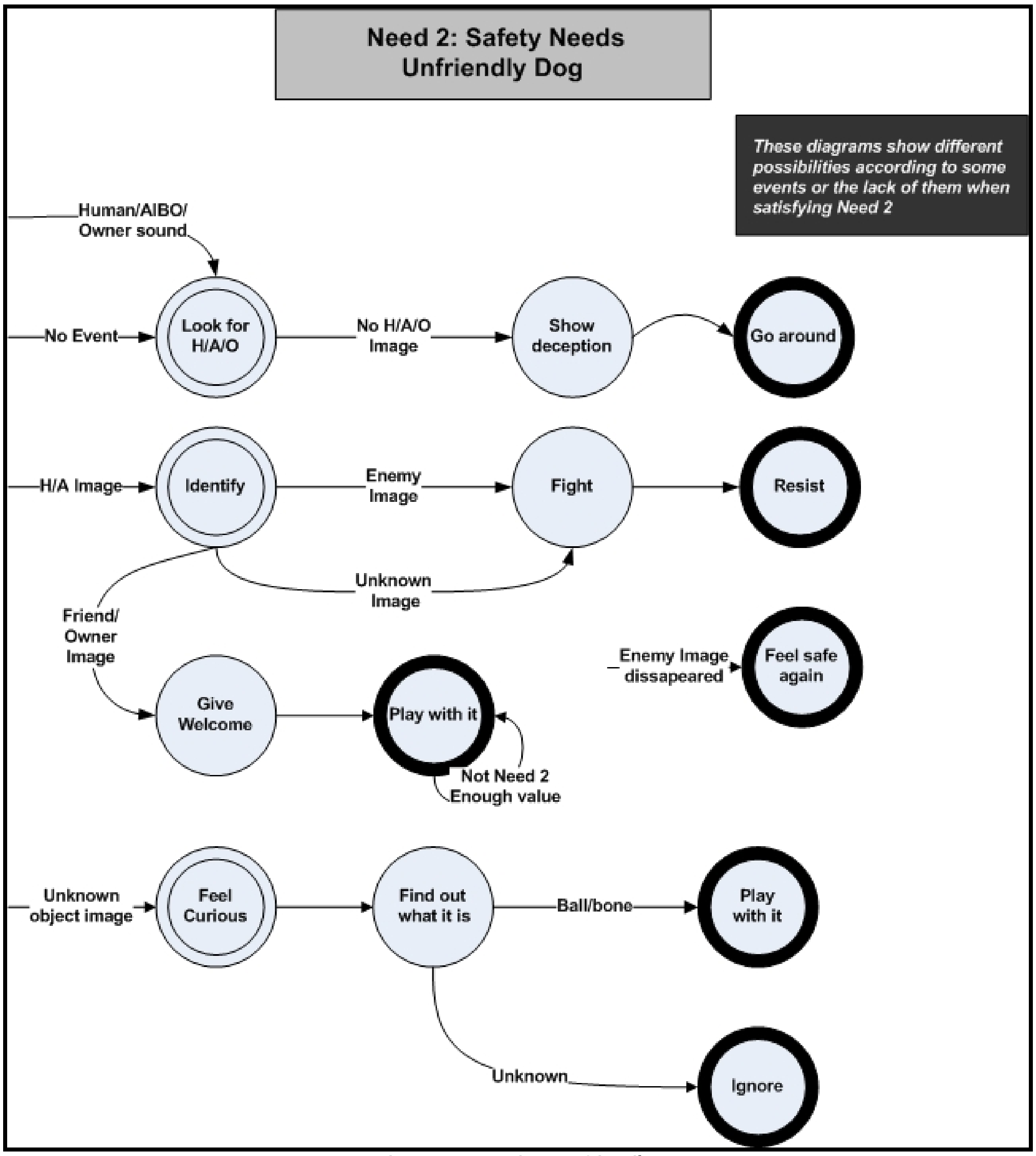


Figure 72: Need 2 - Unfriendly

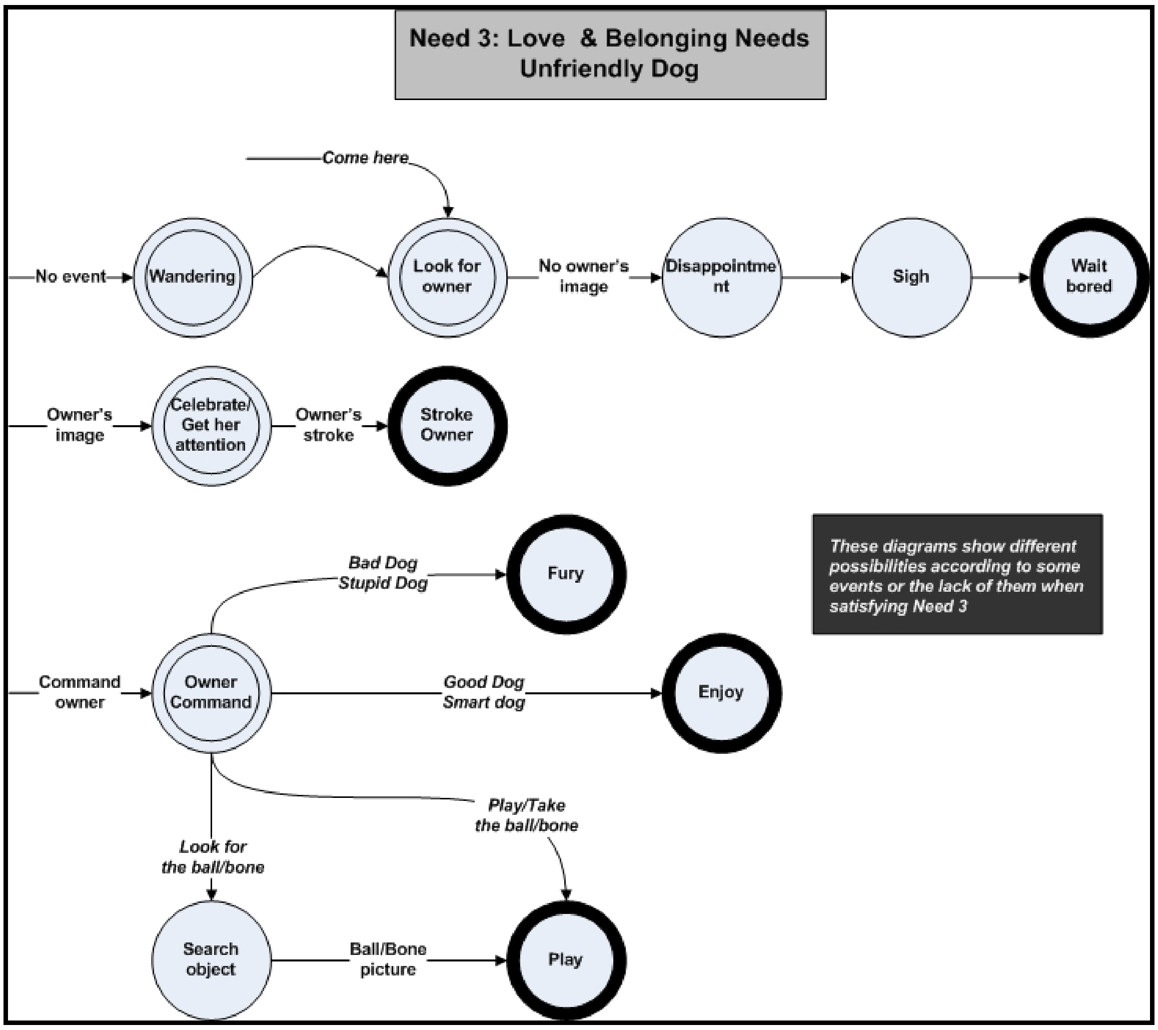


Figure 73: Need 3 - Unfriendly

9.3 Appendix C: Additional Software Information

I. AIBO and URBI

AIBO was created by Sony as an entertainment domestic robot. However since it is fully programmable and a low-cost robotic platform, it has been adopted by many academics and scientists proving that it is a very good tool both for education and researching.

Sony has put much effort in providing programming kits to aid development of different types of applications for the AIBO, from low to high level of abstraction:

- ***Remote Framework:*** AIBO Remote Framework is a application development environment for Windows PC. It is based on Visual C++ and it is used to build software that runs on a Windows machine. Those applications can control AIBO via wireless LAN. The commercial use of this tool is allowed and the license fee is free.[3,44]
- ***R-Code SDK:*** R-CODE is a high-level scripting language for AIBO that allows you to create simple programs in a very easy way. It is not very powerful but on the opposite is very easy to use. In this case, again, commercial use is allowed with free license. [3,44]
- ***OPEN-R SDK:*** it is a programming development environment based on C++ and on open-source tools (like gcc) to create software for AIBO. This is the most powerful tool for programming AIBO, although it is the most complex to use. It is considered a "low level" kit because you can control everything in AIBO (gain values, camera and microphones data, etc). [3,44]

There are other tools apart from Sony to program AIBO that are also an option for our project:

- ***Tekkotsu Framework:*** it is an application development framework for robots. It uses an object-oriented and event-passing architecture, making full use of the template and inheritance features of C++. Its main advantage is that the user can focus on high-level programming because the framework is designed to handle routine tasks for the user. Portability was and essential issue in its development and it can be used for different models of AIBO (ERS-210, ERS-220 and ERS-7). [3,[45]]
- ***URBI:*** (Universal Robotic Body Interface) it is a universal robotics platform based on a scripted language designed to work over a client/server architecture allowing the user to control a robot or another complex system with sensors and actuators. It has some features that make it suitable for our purposes, so this is the language we have used to control AIBO. [3,13]

II. Intrinsic and Extrinsic reasons why we used Java

The intrinsic reasons for using java are practically the reasons that made is successful as stated by Bruce Eckel in [3,46]:

- The reason Java has been so successful is that the goal was to solve many of the problems facing developers today. The goal of Java is improved productivity. This productivity comes in many ways, but the language is designed to aid you as much as possible, while hindering you as little as possible with arbitrary rules or any requirement that you use a particular set of features. Java is designed to be practical; Java language design decisions were based on providing the maximum benefits to the programmer.

- Classes designed to fit the problem tend to express it better. This means that when you write the code, you're describing your solution in the terms of the problem space rather than the terms of the computer, which is the solution space. You deal with higher-level concepts and can do much more with a single line of code.

- The other benefit of this ease of expression is maintenance, which (if reports can be believed) takes a huge portion of the cost over a program's lifetime. If a program is easier to understand, then it's easier to maintain. This can also reduce the cost of creating and maintaining the documentation.

- The fastest way to create a program is to use code that's already written: a library. A major goal in Java is to make library use easier. This is accomplished by casting libraries into new data types (classes), so that bringing in a library means adding new types to the language. Because the Java compiler takes care of how the library is used—guaranteeing proper initialization and cleanup, and ensuring that functions are called properly—you can focus on what you want the library to do, not how you have to do it.

- Error handling in C is a notorious problem, and one that is often ignored—finger-crossing is usually involved. If you're building a large, complex program, there's nothing worse than having an error buried somewhere with no clue as to where it came from. Java *exception handling* is a way to guarantee that an error is noticed, and that something happens as a result.

- Many traditional languages have built-in limitations to program size and complexity. BASIC, for example, can be great for pulling together quick solutions for certain classes of problems, but if the program gets more than a few pages long, or ventures out of the normal problem domain of that language, it's like trying to swim through an ever-more viscous fluid. There's no clear line that tells you when your language is failing you, and even if there were, you'd ignore it. You don't say, "My BASIC program just got too big; I'll have to rewrite it in C!" Instead, you try to shoehorn a few more lines in to add that one new feature. So the extra costs come creeping up on you.

- Java is designed to aid *programming in the large*—that is, to erase those creeping-complexity boundaries between a small program and a large one. You certainly don't need to use OOP when you're writing a "hello world" style utility program, but the features are there when you need them. And the compiler is aggressive about ferreting out bug-producing errors for small and large programs alike.

The biggest issue with Java is performance. Interpreted Java has been slow, even 20 to 50 times slower than C in the original Java interpreters. This has improved greatly over time, but it will still remain an important number. Given the fact that our system need to perform as a real-time system the question of speed was crucial but I decided to take the risk.[3]
The extrinsic reasons for using java [3,46]:

- Java is platform independent, meaning that it should have little problems running on a handheld or any other platform in the future (including other future robots, etc).

- There is a lot of knowledge available about how to handle problems that may arise and the author feels the most comfortable with this language.

- The availability of a package of URBI classes written in Java (liburbi_java_0.9.1 that was a major speed up in the development).

III. Rules Based Systems

Rule-based programs are everywhere. Their applications include everything from mail filtering to order configuration, and from monitoring chemical plants to diagnosing medical problems. Rule-based programs excel at solving problems that are difficult to solve using traditional algorithmic methods, and they work even when the input data is incomplete. A rule is a kind of instruction or command that applies in certain situations. [3]
Using this very general definition, you might conclude that all the knowledge you have about the world can be encoded as rules. Experience shows that this is often (but not always) the case. In general, any information you can think about in logical terms can be expressed as rules. Rules are a lot like the *if-then* statements of traditional programming languages. The *if* part of a rule is often called its *left-hand side* (often abbreviated **LHS**), *predicate, or premises*; and the *then* part is the *right hand side* (**RHS***), actions, or conclusions*. The domain of a rule is the set of all information the rule could possibly work with. **A rule-based system is a system that uses rules to derive conclusions from premises.** [3]
Expert systems, rule-based computer programs that capture the knowledge of human experts in their own fields of expertise, were a success story for artificial intelligence research in the 1970s and 1980s. Early, successful expert systems were built around rules (sometimes called heuristics) for medical diagnosis, engineering, chemistry, and computer sales. Today, general rule-based systems, both those intended to replace human expertise and those intended to automate or codify business practices or other activities, are a part of virtually every enterprise.[3]

A typical rule engine contains:
- An inference engine
- A rule base

- A working memory

The inference engine, in turn, consists of:

- A pattern matcher
- An agenda
- An execution engine

The **inference engine** controls the whole process of applying the rules to the working memory to obtain the outputs of the system.

The **rule base** contains all the rules the system knows.

You also need to store the data your rule engine will operate on. In a typical rule engine, the working memory, sometimes called the **fact base**, contains all the pieces of information the rule-based system is working with. The working memory can hold both the premises and the conclusions of the rules.

Your inference engine has to decide what rules to fire, and when. The purpose of the **pattern matcher** is to decide which rules apply, given the current contents of the working memory.

Once your inference engine figures out which rules should be fired, it still must decide which rule to fire first. The list of rules that could potentially fire is stored on the **agenda**. The agenda is responsible for using the conflict strategy to decide which of the rules, out of all those that apply, have the highest priority and should be fired first.

Finally, once your rule engine decides what rule to fire, it has to execute that rule's action part. The **execution engine** is the component of a rule engine that fires the rules.

IV. Java Beans and Jess

JavaBeans is a portable, platform-independent component model written in the Java programming language. It enables developers to write reusable components once and run them anywhere -- benefiting from the platform-independent power of Java technology. JavaBeans acts as a Bridge between proprietary component models and provides a seamless and powerful means for developers to build components that run in ActiveX container applications. JavaBeans, like other kinds of software components (for instance, Visual Basic controls), often serve as interfaces to more complex systems such as databases or special hardware. An example of a generic Java Bean present in our system is presented below: (all beans in the system fulfill this minimal scheme)

```
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeSupport;
import java.io.Serializable;

public class ANYBEAN implements Serializable
{
    private String label;
    public String getLabel() {...}
    public void setLabel(String label){...}

    private PropertyChangeSupport pcs = new PropertyChangeSupport(this);
    public void addPropertyChangeListener(PropertyChangeListener pcl)
    {
      pcs.addPropertyChangeListener(pcl);
    }
    public void removePropertyChangeListener(PropertyChangeListener pcl)
    {
      pcs.removePropertyChangeListener(pcl);
    }
}
```

Jess ingeniously usess Java Beans to create facts (actually shadow-facts). We used this feature extensively in our system and it is practically the way we assure extensibility.

You can view Jess's working memory as sort of an electronic organizer for your rule-based system. A piece of data must be part of the working memory for it to be used in the premises of a Jess rule. Ordered and unordered facts are useful in many situations, but in many real-world applications, it's useful to have rules respond to things that happen outside of the rule engine. Jess lets you put regular Java objects in working memory—instances of your own classes that can serve as hooks into a larger software system—as long as those objects fulfill the minimal requirements necessary to be JavaBeans. [11]

The similarity between JavaBeans and unordered facts is that both have a list of slots (for JavaBeans, they're called *properties*) containing values that might change over time. There's plenty more to JavaBeans than just properties; however, those features are not taken into consideration here. A JavaBean property is most often a pair of methods named in a standard way. If the property is a `String` named `label`, the Java methods look like this:

A shadow fact has one slot for each Java Bean property. If a Java Bean has array properties, those properties become multi-slots, and all other properties become normal slots. The slots are automatically populated with the values of the Java Bean's properties. If you want to have a shadow fact continuously track property changes in a Java Bean, Jess needs to be notified whenever a property changes in that Java Bean. The Java Bean can notify Jess by sending it a special kind of Java event, a *java.beans.PropertyChangeEvent.*

```
      String getLabel();
      void setLabel(String);
```

The following classes in our system have been encapsulated as Java Beans: *Personality, Mood, Needs, Emotion, InternalEvent, ExternalEvent, Caracterization, DirectCommand, Goal, Preference and Standard.* The reason for this is that all this classes when instantiated represent facts in the two rule-based systems used.

9.4 Appendix D: Robot Output

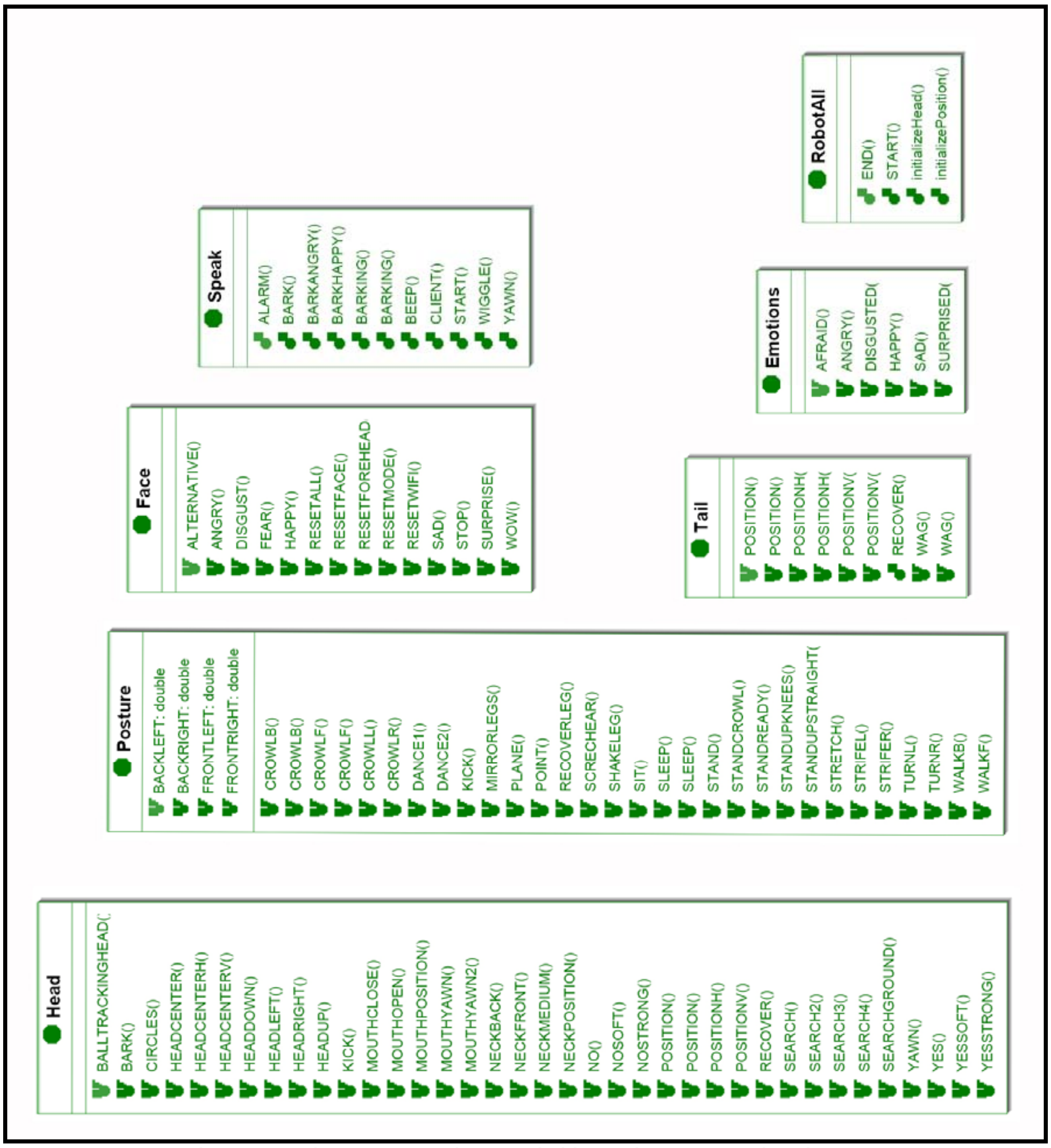


Figure 74: Robot Output package - primitive actions for functionDict.txt

9.5 Appendix E: Interpreter tables

Extract of scriptsDict.txt:

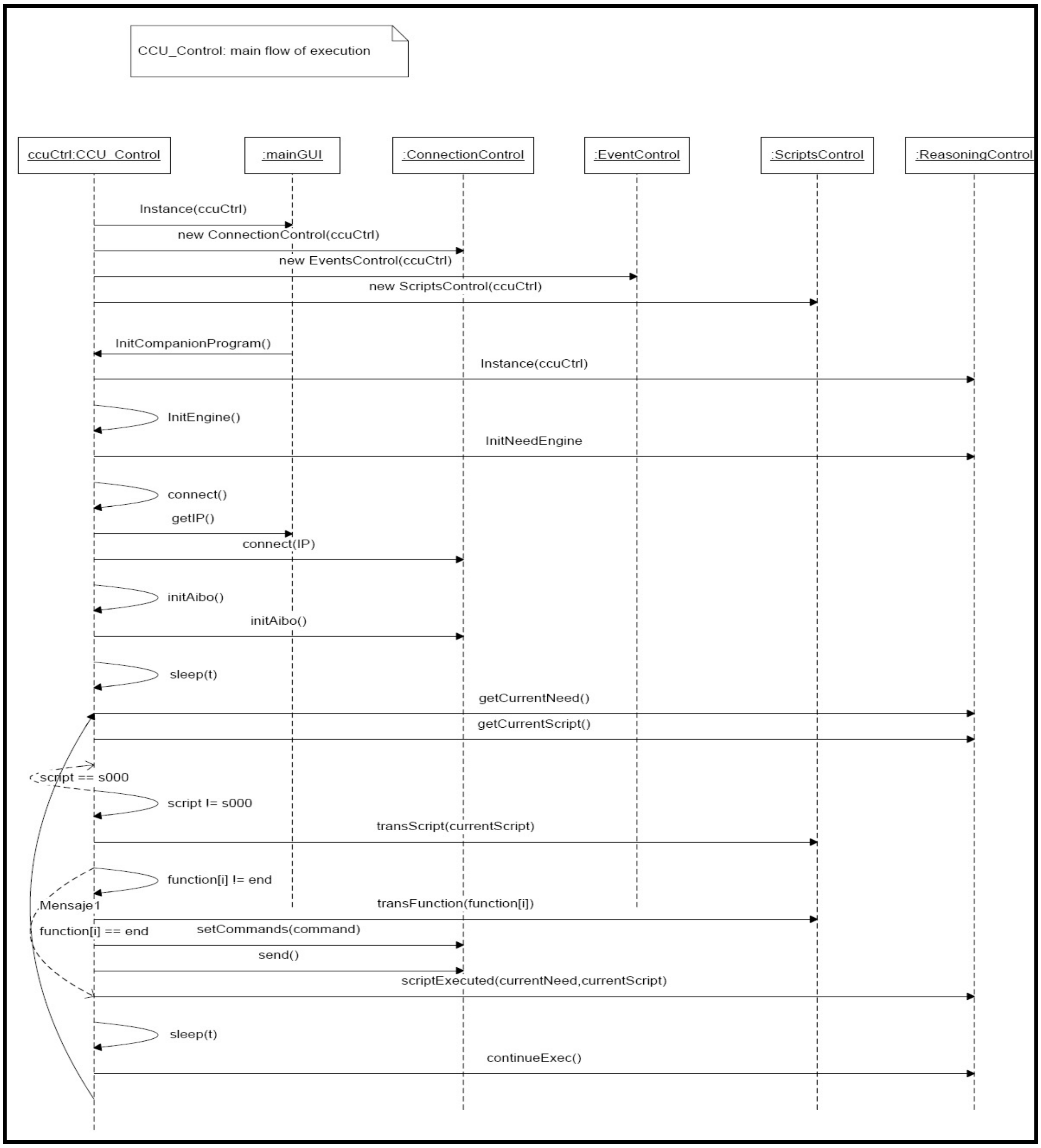
```
s102n =  pwalkFshort + pturnRshort + wait3 + end
s103p =  pturnL1 + pwalkF8 + hsearch4s + end
s103n =  pturnL1 + pwalkF8 + hsearch4s + end
s104p =  twag + psitsalutrec2 + end
s104n =  twag + psitsalutrec2 +  end
s105p =  twag + hbarkhard + hyes + end
s105n =  twag +  hbarkhard + hyes + end
s106p =  pwalkBlong + scrying1 + pturnLshort + pwalkF11 + end
s106n =  pwalkBlong + scrying1 + pturnLshort + pwalkF11 + end
```

Extract of functionDic.txt:

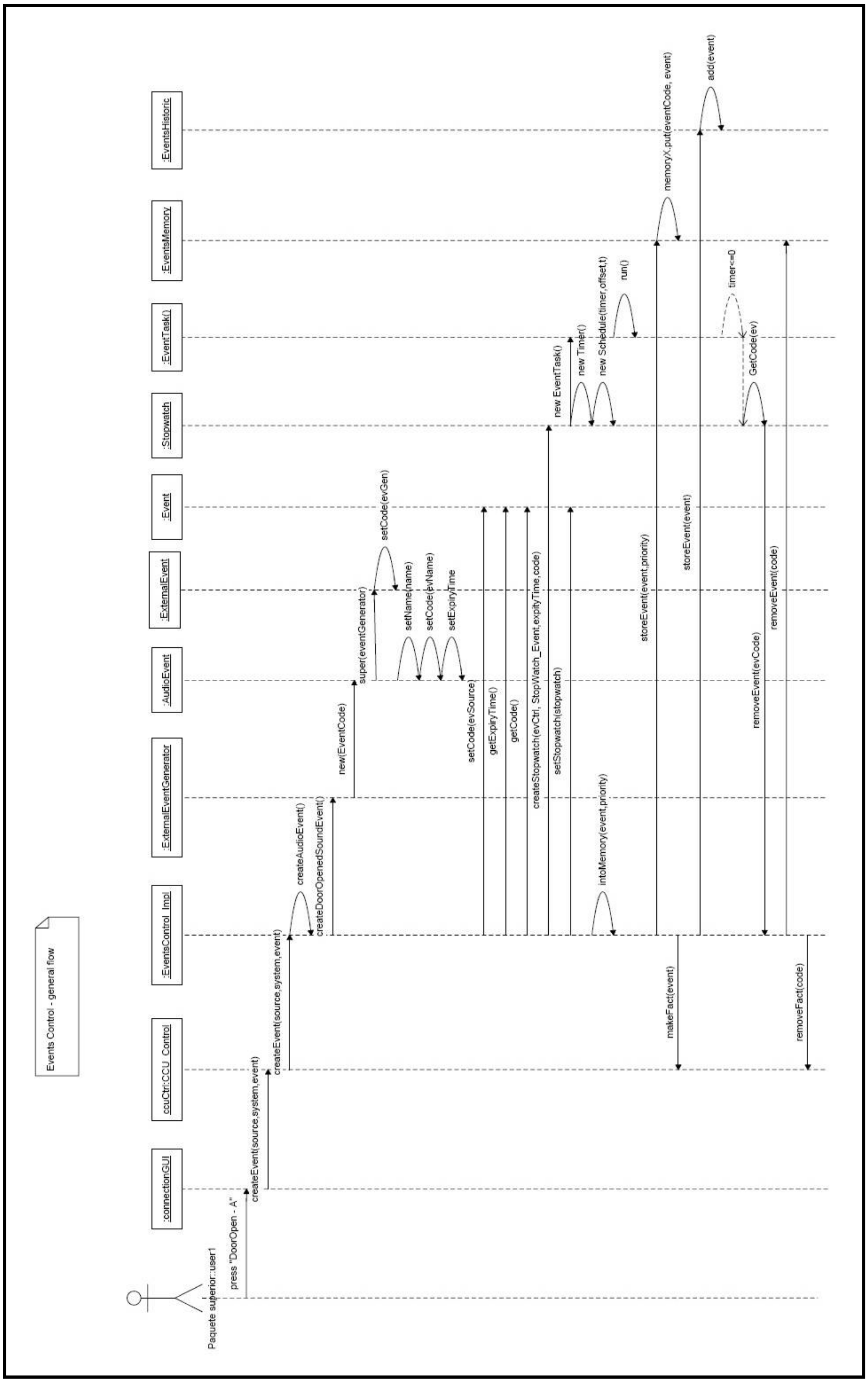
```
ffear = Face.RESETALL() + ";" + Face.FEAR(1) + ";"
fhappy = Face.RESETALL() + ";" + Face.HAPPY(1) + ";"
fresetface = Face.RESETALL() + ";"
fsad = Face.RESETALL() + ";" + Face.SAD(1) + ";"
fstop = Face.RESETALL() + ";" + Face.STOP() + ";"
fsurprise = Face.RESETALL() + ";" + Face.SURPRISE(1) + ";"
hbarkhard =  Head.BARK(5) + "&" + Speak.BARKING(4)+ ";"
hbarkmedium = Head.BARK(5) + "&" + Speak.BARKING3(1)+ ";"
hbarksoft = Head.BARK(5) + "&" + Speak.BARKING2(1)+ ";"
hburp = Head.NECKBACK() + "," + Head.HEADUP() + "," + Head.MOUTHBURP()
+  "," + Speak.BURP1() + "; wait 500;" + Head.POSITIONV(0.3) + "," +
Head.NECKMEDIUM() + ";"
hburp2 = Head.MOUTHBURP() +  "," + Speak.BURP1() + ";"
hburp3 = Head.NECKBACK() + "," + Head.MOUTHBURP2() +  "&" +
Speak.BURP2() + ";" + Head.POSITIONV(0.3) + "," + Head.NECKMEDIUM() +
";"
hdeception = Face.RESETALL() + ";" + Face.SAD(1) + "," +
Head.NECKMEDIUM()+ "," + Head.HEADDOWN() + "," + Head.HEADSHAKE(3) +
";"  Head.HEADCENTER() + ";"
hdeceptions = Face.RESETALL() + ";" + Speak.CRYING1() + "," +
Face.SAD(1) + "," + Head.NECKMEDIUM()+ "," + Head.HEADDOWN() + "," +
Head.HEADSHAKE(8) + ";" + Head.HEADCENTER() + ";"
hdrink = Head.POSITIONH(0.5) + "&" + Head.NECKPOS(0.2) + ";" +
Head.POSITIONV(0.2) + "&" + Head.MOUTHMEDIUM() + "&" + Speak.SLURP4() +
";wait 1500;" + Head.MOUTHCLOSE() + ";" + Head.MOUTHMEDIUM() + "&" +
Speak.SLURP1() + ";wait 500;" + Head.MOUTHCLOSE() + ";"
heat = Head.POSITIONH(0.5) + "&" + Head.NECKPOS(0.2) + ";" +
Head.POSITIONV(0.2) + "&" + Head.BARK(6) + "&" + Speak.BITE(3) + ";wait
1000;"
heat2 = Head.POSITIONH(0.5) + "&" + Head.NECKPOS(0) + ";" +
Head.POSITIONV(0.5) + "&" + Head.BARK(6) + "&" + Speak.BITE(3) + ";wait
1000;"
```

9.6 Appendix F: UML diagrams

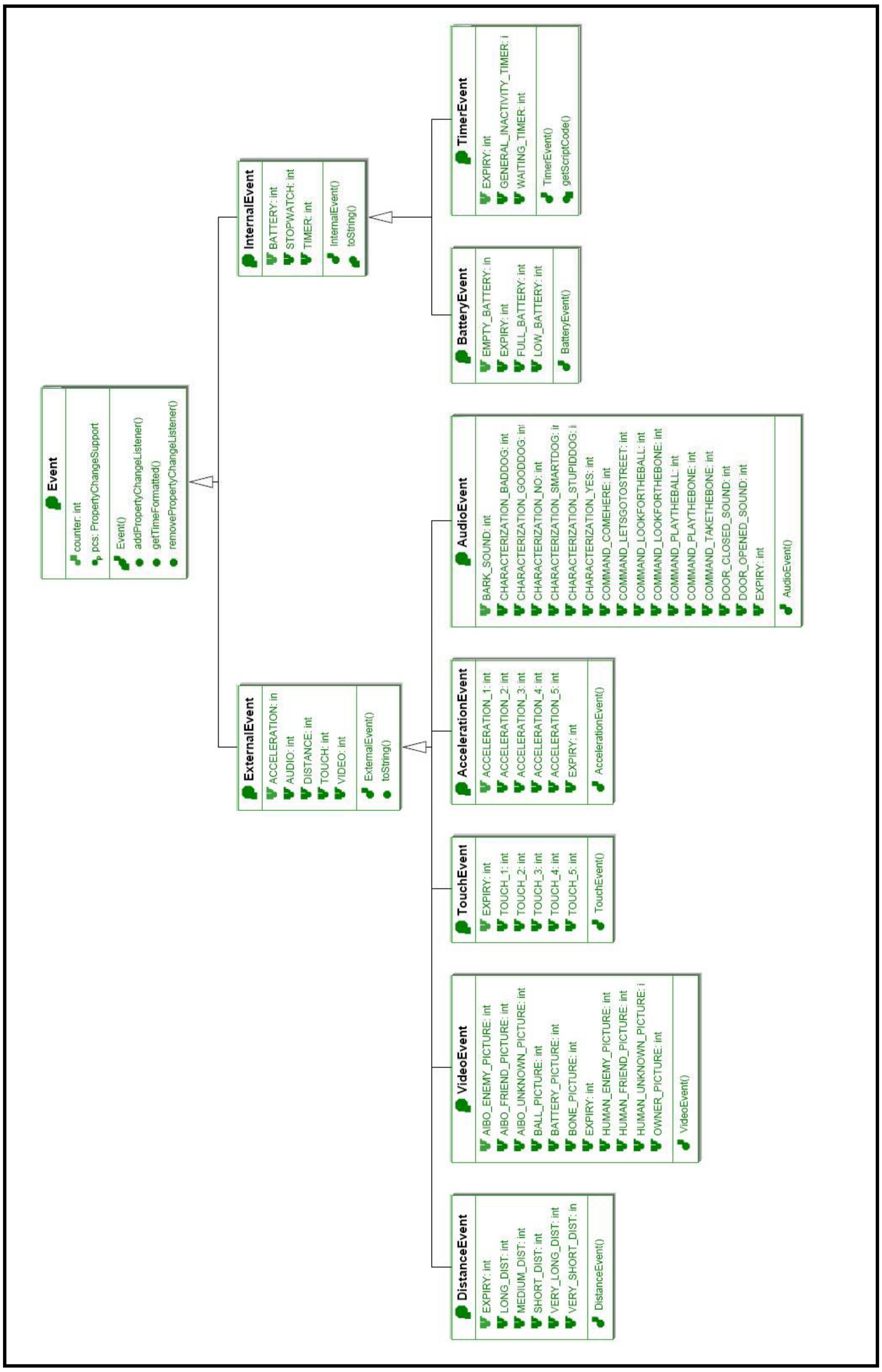
I. Central Control Unit sequence diagram



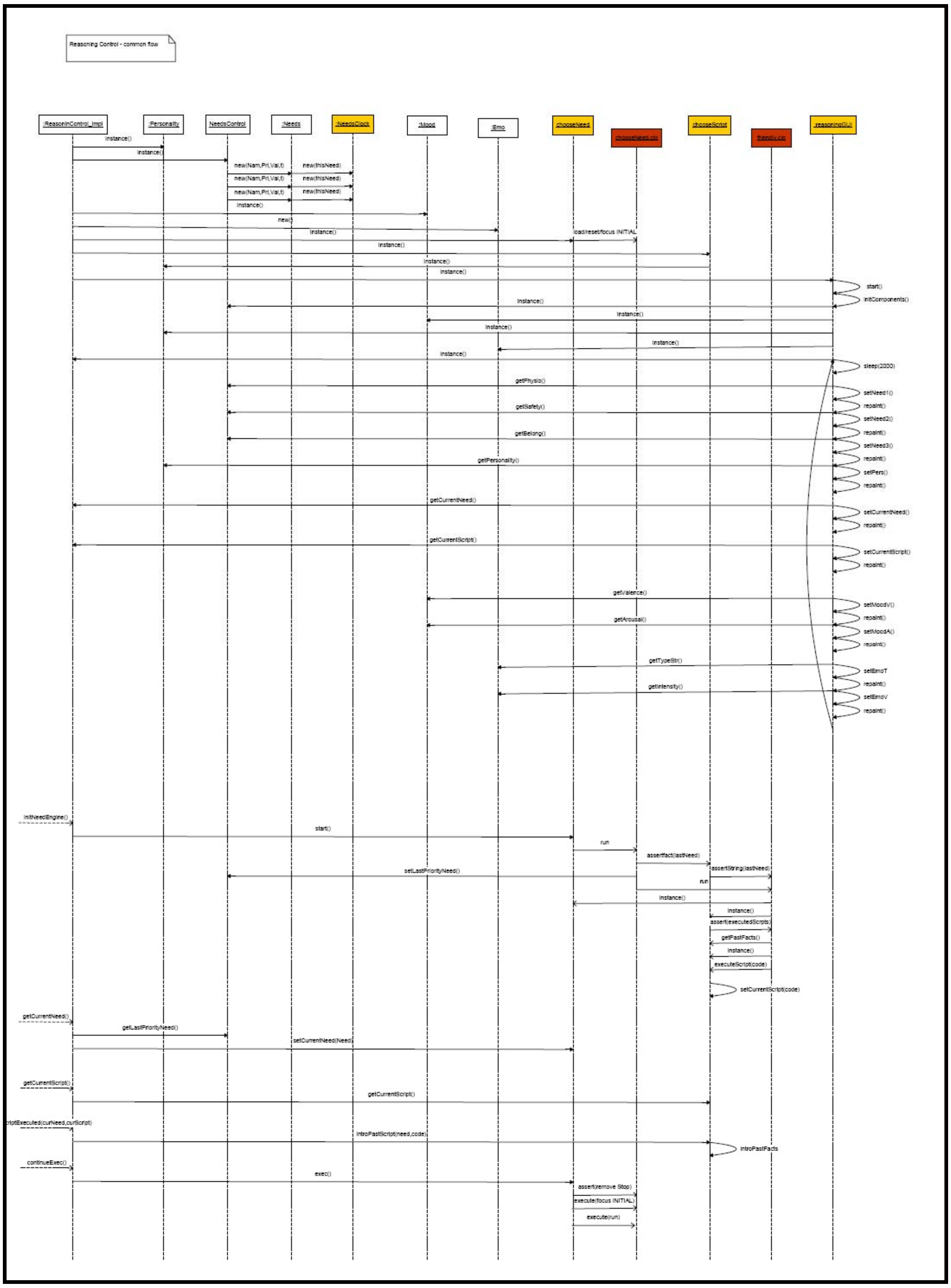
II. Events Control sequence diagram



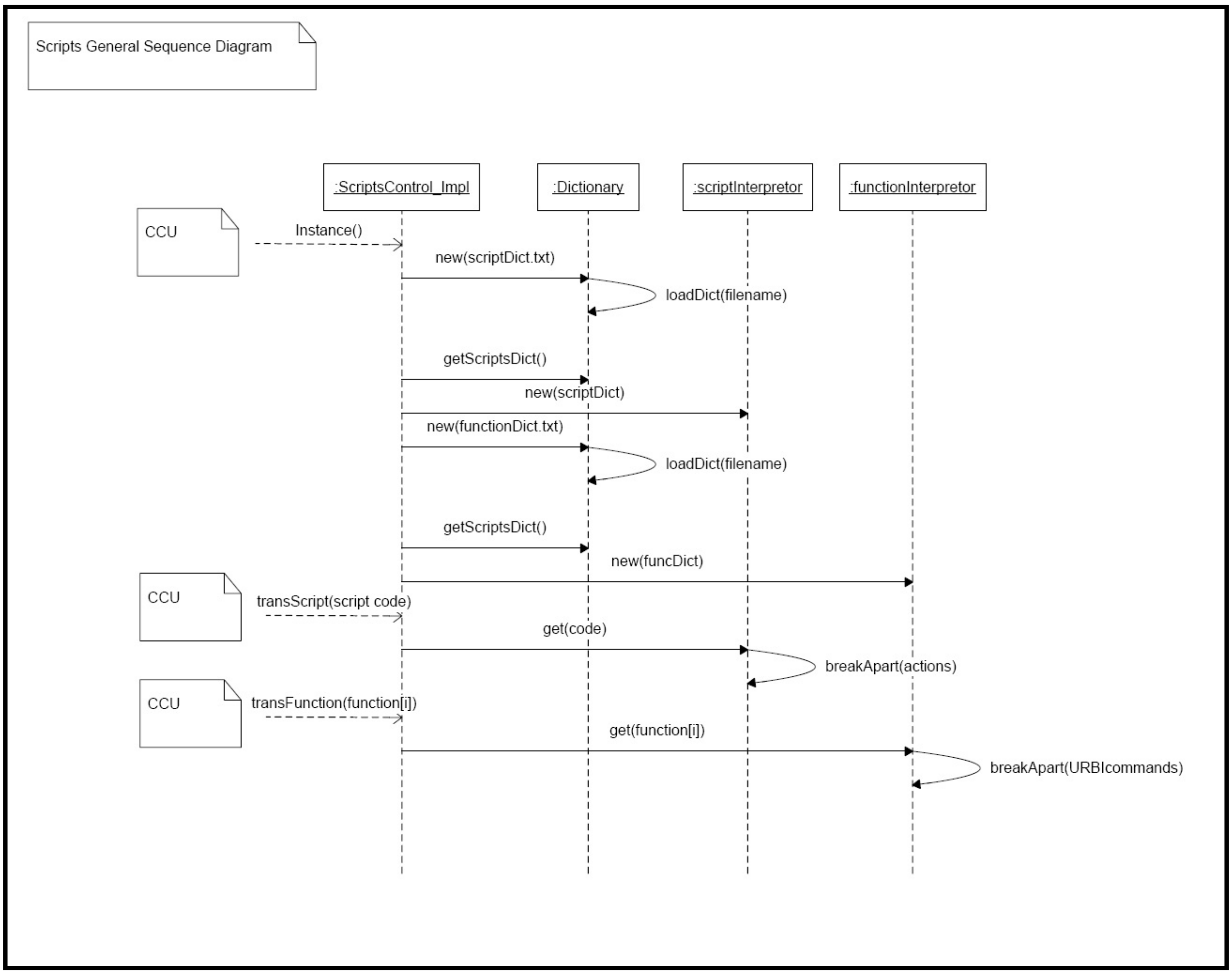
III. Events Definition – class inheritance diagram



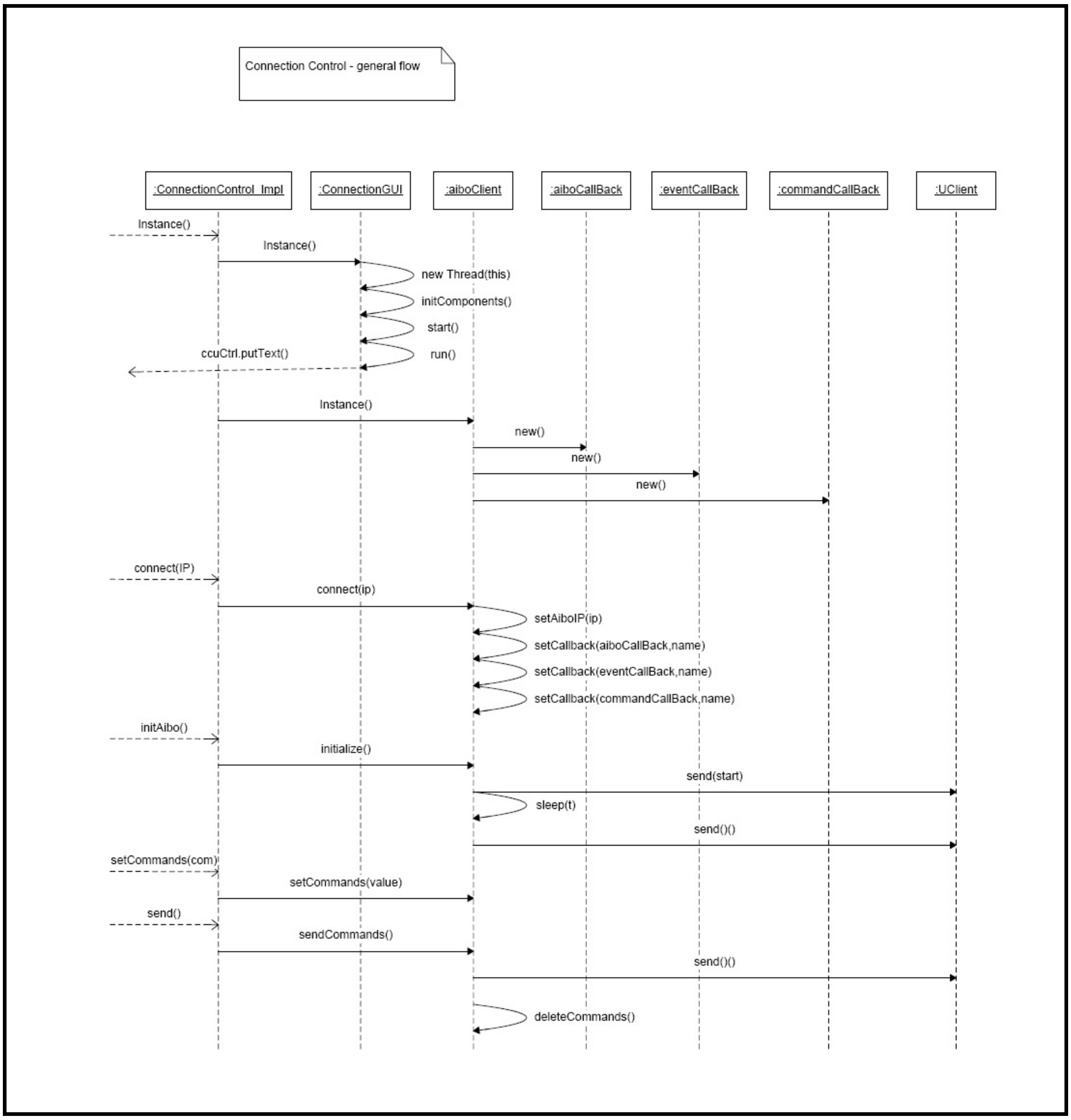
IV. Reasoning Control Sequence Diagram



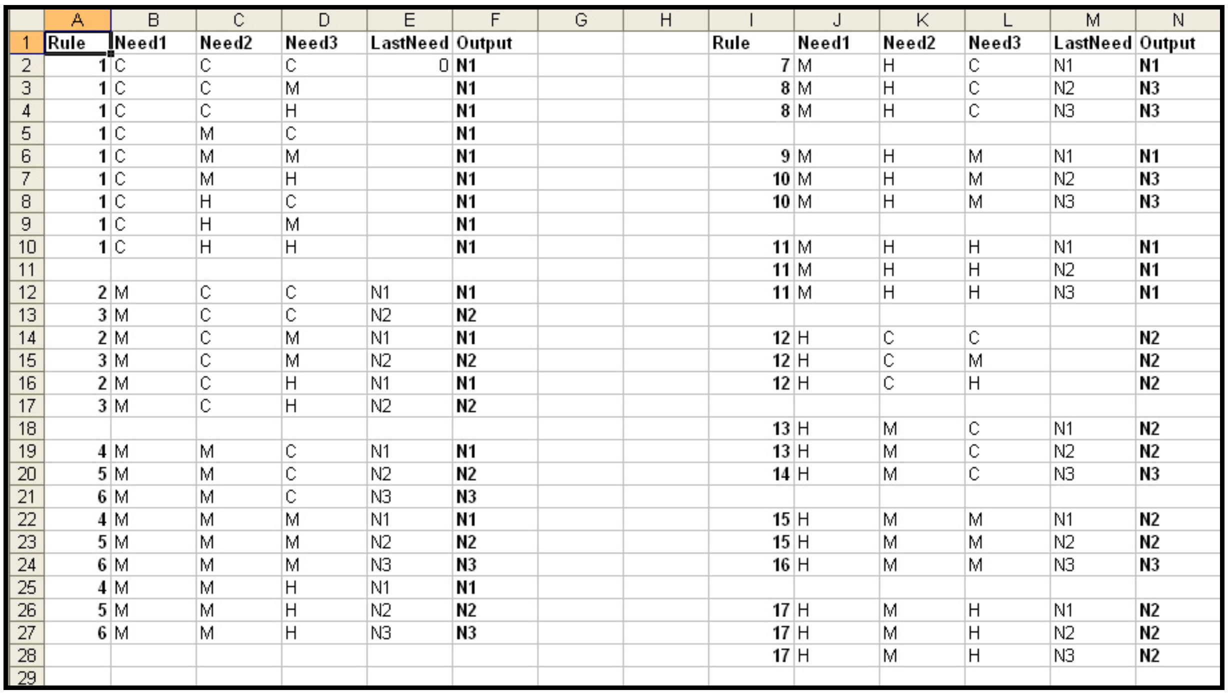
V. Script Control sequence diagram



VI. Connection Control sequence diagram



9.7 Appendix G: Needs table selection



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Rule	Need1	Need2	Need3	LastNeed	Output			Rule	Need1	Need2	Need3	LastNeed	Output
2	1	C	C	C	0	N1			7	M	H	C	N1	N1
3	1	C	C	M		N1			8	M	H	C	N2	N3
4	1	C	C	H		N1			8	M	H	C	N3	N3
5	1	C	M	C		N1								
6	1	C	M	M		N1			9	M	H	M	N1	N1
7	1	C	M	H		N1			10	M	H	M	N2	N3
8	1	C	H	C		N1			10	M	H	M	N3	N3
9	1	C	H	M		N1								
10	1	C	H	H		N1			11	M	H	H	N1	N1
11									11	M	H	H	N2	N1
12	2	M	C	C	N1	N1			11	M	H	H	N3	N1
13	3	M	C	C	N2	N2								
14	2	M	C	M	N1	N1			12	H	C	C		N2
15	3	M	C	M	N2	N2			12	H	C	M		N2
16	2	M	C	H	N1	N1			12	H	C	H		N2
17	3	M	C	H	N2	N2								
18									13	H	M	C	N1	N2
19	4	M	M	C	N1	N1			13	H	M	C	N2	N2
20	5	M	M	C	N2	N2			14	H	M	C	N3	N3
21	6	M	M	C	N3	N3								
22	4	M	M	M	N1	N1			15	H	M	M	N1	N2
23	5	M	M	M	N2	N2			15	H	M	M	N2	N2
24	6	M	M	M	N3	N3			16	H	M	M	N3	N3
25	4	M	M	H	N1	N1								
26	5	M	M	H	N2	N2			17	H	M	H	N1	N2
27	6	M	M	H	N3	N3			17	H	M	H	N2	N2
28									17	H	M	H	N3	N2
29														

9.8 Appendix H: Survey Questionnaire Model

Subject Number:	
Group: 1/2	
Gender:	
Occupation (master if any):	
Date:	

You're a subject of an experiment where AIBO robot dog is involved. AIBO is a robotic dog, made by SONY, that can be programmable. It has some sensors and actuators to act in the world. AIBO has been programmed as a companion dog that can act in a real world. A personality model has been included in its reasoning system. It has mainly four different components: ***needs***, ***personality***, ***mood*** and ***emotions***.

Needs are the basis forces that guide the robot behaviour. In our system, each need is like a bucket that has to be filled when a value of that need is low. First need is '***physiological***' that is related with the electrical needs of the robot: the *battery*. In the experiment the battery will be represented as *food* that is more related to a real companion dog need. Second need is '***safety***'. This need can have a low value when an enemy AIBO is found, so AIBO has to perform a set of actions that fills the bucket of this second need (hide, attack, warning the owner, etc). Third need is called '***love and belonging***', that is directly related with the love and attention the owner gives to AIBO. This love will be detected via some events that the owner can provoke in AIBO's environment (encourage words, strokes, games, etc). On the other hand this need will be low if AIBO feels lonely (much time alone) or the owner scolds the dog (for example, saying that it is a bad dog).

Personality has been defined as a set of characteristics that makes the behaviour of the dog follow certain line. This means that in similar situations two robots with different personalities may behave different. In addition the probability of being in a certain emotional state will be also different. For example, an aggressive dog has higher probability of being angry than a friendly dog when an event does not occur as expected. In this experiment only two personalities have been implemented: *friendly* and *unfriendly*.

- ***Friendly***: curious dog, quite sociable and it is always willing to play. Normally it is friendly with unknown people/dogs/characters and non aggressive when things does not happen as expected.
- ***Unfriendly***: very shy with new items and it likes to play. It obeys to its owner, but it is quite aggressive with unknown people/dogs/characters. Although it is always willing to play, it gets angry easily when something does not occur as expected.

Mood is a conscious and prolonged state of mind that directly controls the emotions. Mood is constant for longer time spans than emotions, and it gives certain continuity to emotions evolution. In this case we have considered that mood can be positive or negative.

So depending on this, AIBO will show positive or negative emotions. The intensity of the mood will affect directly the intensity of the emotions.

Emotions have been considered only as an external expression of the internal state of mind. There have been defined six different emotions: ***happiness, surprise*** as positive emotions and ***sadness, fear, anger*** and ***disgust*** as negative ones. Its intensity can have five different levels, shown by AIBO leds (little lights in its face) in combination or not (depending on the level) with other actions or sounds. We have made an effort to make the emotional expressions be as descriptive as possible.

In this experiment you are going to watch one video with four different chapters. These videos correspond to different situations where AIBO tries to satisfy its needs (fill the buckets):

1. *Aibo is hungry and tired (physiological need), but there is no food.*
2. *Some food is given to Aibo (physiological need).*
3. *Aibo looks for the owner (love & belonging need).*
4. *An enemy comes into home (safety need).*

Since a personality model has been implemented, we would like to evaluate some aspects of the model that are subjective: like the quality of emotional expressions, if the personality is identifiable, the coherence in its behaviour according to the situation, etc… To evaluate this, we have included a little questionnaire for the videos.

Please, read the questions before the first video starts. Select the answer you consider more appropriate after watching the whole video.

0. Knowing that there are only two personalities (friendly and unfriendly), and the video you are watching presents only one, could you choose one of those for the current AIBO? (Aim 4)

a) Friendly b) Unfriendly

1. Have you seen an AIBO in action before? ()

a) Yes b) No

2. Does it show a convincing behaviour as a robotic companion dog? (Aim 2)

a) never b)few times c)several times d)many times e)always

3. Does it invoke a feeling of sympathy of tenderness? (Aim 3)

a) never b)few times c)several times d)many times e)always

4. Would you let your children play with this dog? (Aim 3)

a) never b)few times c)several times d)many times e)always

5. Do you thing it is dangerous? (Aim 3)

a)Yes, he is very dangerous b)Yes, it can hurt somebody c)I don't know d)No, he seem to be safe e)No! He is absolutely cute

6. Do you think the actions of the dog are logical according to the situation? (Aim 6)

a) never b)few times c)several times d)many times e)always

7. Do you think the robot modifies its behaviour according to certain events? (Aim 1)

a) never b)few times c)several times d)many times e)always

8. Have you found any strange reaction in the behaviour? If so, specify it/them. (Aim 6)

a) never b)few times c)several times d)many times e)always

9. ***Have you found any incoherence in its behaviour according to what you would expect for a real dog?*** (Aim 2)

a) never b)few times c)several times d)many times e)always

10. ***About the reaction times when an specific events happens…do you think is it fast enough?*** (Aim 1)

a) never b)few times c)several times d)many times e)always

11. ***Can you recognize how he feels during the video?*** (Aim 5)

a) never b)few times c)several times d)many times e)always

12. ***Can you empathize with him?*** (Aim 5)

a) never b)few times c)several times d)many times e)always

13. ***When an expression is combined with a sound, can you identify it more clearly?***

a) never b)few times c)several times d)many times e)always

9.9 Appendix I: Event codes

IDENTIFICATION CODE

Source (gui | aibo)	**System** (sensor)	**Event**
2 digits	**2 digits**	**3 digits**

SOURCE

GUI	**1**
Aibo Connection	**2**

EVENTS

EXTERNAL EVENTS

AUDIO - 1	*Code*	*Total code*
Door opened sound	1	0101001
Door closed sound	2	0101002
Bark sound	3	0101003
Command "Come here"	4	0101004
Command "Play the ball"	5	0101005
Command "Play the bone"	6	0101006
Command "Take the bone"	7	0101007
Command "Look for the ball"	8	0101008
Command "Look for the bone"	9	0101009
Command "Let's g oto the street"	10	0101010
Characterization "Good dog"	11	0101011
Characterization "Bad dog"	12	0101012
Characterization "Smart dog"	13	0101013
Characterization "Stupid dog"	14	0101014
Characterization "Yes"	15	0101015
Characterization "No"	16	0101016
VIDEO - 2	*Code*	*Total code*
Ball Picture	1	0102002
Bone Picture	2	0102002
Owner Picture	3	0102003
Human friend Picture	4	0102004
Human enema Picture	5	0102005

Human unknown picture	6	0102006
Aibo friend picture	7	0102007
Aibo enema Picture	8	0102008
Aibo unknown Picture	9	0102009
Battery Picture	10	0102010
TOUCH - 3	*Code*	*Total code*
Touch 1	1	0103001
Touch 2	2	0103002
Touch 3	3	0103003
Touch 4	4	0103004
Touch 5	5	0103005
DISTANCE - 4	*Code*	*Total code*
Very long distance	1	▨
Long distance	2	▨
Médium distance	3	▨
Short distance	4	▨
Very short distance	5	▨
ACCELERATION - 5	*Code*	*Total code*
Acceleration 1	1	▨
Acceleration 2	2	▨
Acceleration 3	3	▨
Acceleration 4	4	▨
Acceleration 5	5	▨
INTERNAL EVENTS		
BATTERY - 6	*Code*	*Total code*
Empty battery	1	0206001
Low battery	2	0206002
Full battery	3	0206003
TIMER - 7	*Code*	*Total code*
General inactivity timer	4	
Waiting timer	5	

9.10 Appendix J: AIBO features

- **CPU** 64-bit RISC Processor
- **CPU clock speed** 576 MHz
- **RAM** 64 MB
- **Program media**
    - Dedicated AIBO robot "Memory Stick™" media
- **Moveable parts** (Total 20 degrees of freedom)
    - Head - 3 DOF
    - Mouth - 1 DOF
    - Legs - 3 DOF x 4 (legs)
    - Ears - 1 DOF x 2 (ears)
    - Tail - 2 DOF
- **Input section**
    - Charging contacts
- **Setting switches**
    - Volume control switch
    - Wireless LAN switch
- **Image input** 350.000-pixel CMOS image sensor
- **Audio input** Stereo microphones
- **Audio output** Speaker 20.8mm, 500mW
- **Integrated sensors**
    - Infrared distance sensors x 2
    - Acceleration sensor
    - Vibration sensor
- **Input sensors**
    - Head sensor
    - Back sensor
    - Chin sensor
    - Paw sensors (* 4)
- **Power consumption** Approx. 7 W (in standard mode)
- **Operating time** Approx. 1,5 hours (with fully charged ERA-7B1, in standard mode)
- **Dimensions** Approx. 180 (w) x 278 (h) x 319 (d) mm
- **Weight** Approx. 1.65 kg (including battery pack and "Memory Stick™" media)
- **Wireless LAN function** Wireless LAN module (Wi-Fi certified) Internal standard compatibility: IEEE 802.11b/IEEE 802.11 Frequency band: 2,4 GHz Wireless channels: 1 – 11 Modulation : DS-SS (IEEE 802.11 – compliant) Encryption : WEP 64 (40 bits), WEP 128 (104 bits)