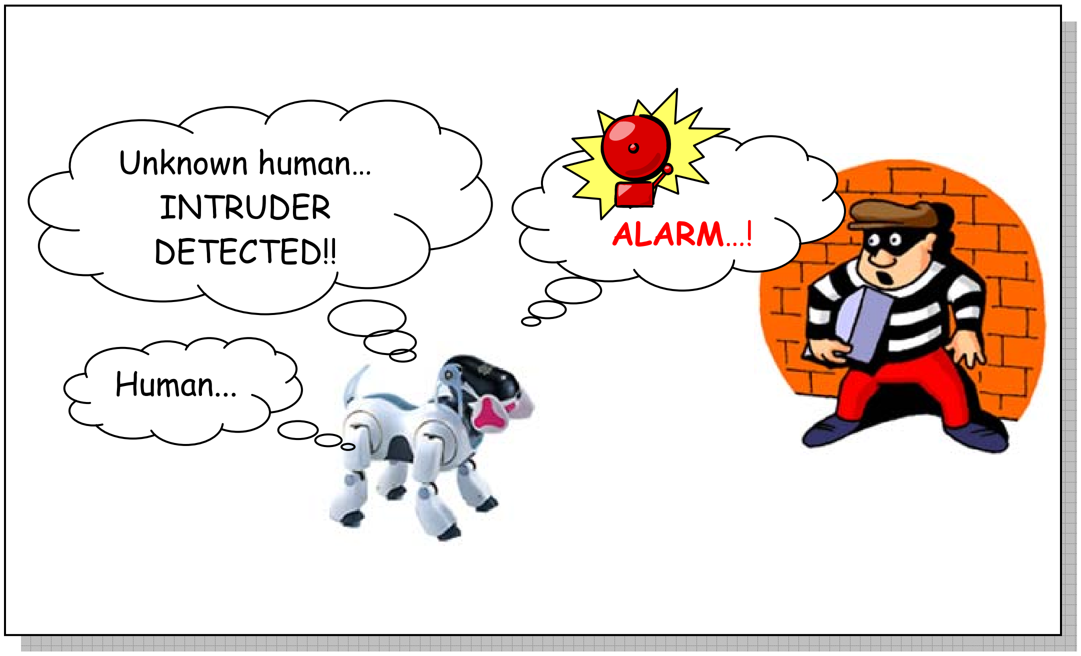
Master Thesis report

AIBO as an Intelligent Robot Watchdog



Bou Tsing Hau
Delft, December 2006

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

TUDelft
Delft University of Technology

Graduation committee:
Dr. drs. L.J.M. Rothkrantz
Ir. H.J.A.M. Geers
Dr. ir. C.A.P.G. van der Mast

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

AIBO as an Intelligent Robot Watchdog
Bou Tsing Hau
Student number: 1015184
E-Mail: boutsing@gmail.com

Title: AIBO as an Intelligent Robot Watchdog
Author: Bou Tsing Hau, 1015184
Discipline: Media & Knowledge Engineering
Delft, December 2006
Graduation committee:
Dr. drs. L.J.M. Rothkrantz
Ir. H.J.A.M. Geers
Dr. ir. C.A.P.G. van der Mast

Abstract

The annual crime reports from the countries in the world have presented us continuous increasing crime affairs. Crime experts are arguing that the increasing of the number of crime affairs is due to the improved registration method. But we are all convinced that an extra security unit can make an environment safer. At least occurred crimes will be registered and extra surveillance in that environment can reduce this problem.

By the rapid advancement of the technology it is nowadays possible to secure the home environment. Static cameras are installed at many places, mounted at the ceiling, to safeguard its environment. The automated panning cameras which are able to turn itself around its axes can monitor a larger area around itself. Since the growing interest of research on image processing, cameras are now also able to recognize moving objects and follow them. The object recognition possibilities of cameras are advancing in a rapid pace, but the problem with static cameras is its limited range. Occluded objects can not be detected by the static cameras. A camera which is able to move around the environment can observe better and more in detail than a static one.

The Sony AIBO robot dog which is able to move itself around freely is a very popular research robot around the world. This Sony AIBO robot has a built-in camera and other modalities which make this robot dog a good approach to free the limitations of the static cameras. This project, AIBO watchdog, has developed a Sony AIBO robot dog to safeguard the home environment. By its camera and microphone evidences can be saved and by using its wireless connection the evidences can be transferred to the right person or instances immediately.

The concept of an AIBO watchdog is not new. The author has developed the AIBO watchdog from the point that the predecessor, Silvia Oana Tanase [40], has left behind. This predecessor only shows the global working of the AIBO watchdog. It lacks of a good fundament which we can use to build the complete one. Therefore it was required to develop a new architecture which has an expansion option to add or replace certain functionalities. The developed AIBO watchdog is intelligent enough to see related events during its patrolling and navigate in the home environment. Furthermore AIBO is able to protect itself from damages during patrolling and carry out appropriate reactions in most situations. To increase its intelligence we have given the AIBO watchdog abilities to prioritize events and use the tools that it has encountered before. The intelligence has been encapsulated in the reasoning system of the AIBO watchdog.

Acknowledgements

First of all, I would like to thank my thesis supervisor, Leon Rothkrantz, for all the inspirations that he gave during my thesis project. His knowledge and ideas have given me a lot of illuminations on my work and many of these became part of my thesis project.

Secondly, I would like to thank my advisors, Siska Fitrianie and Zhenke Yang, for providing me the patient guidance and academic support. Their knowledge and instructions help me to conquer a lot of challenges in the process of my research.

Furthermore, I would like to thank the other PhD. and MSc. students and other people from the university who helped me during the project and gave me useful advices during my thesis project. These are (in a random order): Manou Plattje, Iulia Tatomir, Dragos Datcu, Dimitri Vukadihovic, Gertjan Feijten, Tom Benjamin, Jose Maria Blanco Calvo.

Last, but certainly not least, I would like to thank my parents, my brother, my sister and especially my girlfriend Xueli for supporting me day and night and even in the most difficult times.

TABLE OF CONTENTS

Table of Figures

Table of Tables

1 Introduction

"You don't invent your mission, you detect it."
Victor Frankl

During the last few years violence has not decreased worldwide. In the optimistic way we can say that violence remains approximately constant, but pessimists will tell us violence has increased a few percent a year recently [1][2]. If we take the US as example, the FBI has reported that there were almost 1.4 Million violent crimes in 2005 and more than 10 Million property crimes [2]. Violent crimes can be manslaughter, forcible rape, robbery, or aggravated assault. Examples of property crimes are burglary, larceny-theft, motor vehicle theft, and arson. For the social aspect point of view, if we express these numbers differently, it means that there were at least 3 people shot, raped or robbed every minute and at least 19 people lost their properties in US every minute. These are the results of a country with a population of 300 million. There are 6.5 Billion people on this world [3]. If we extrapolate these numbers, the result will shock us all. Therefore this important social issue concerns the safety of all of us. In order to support security and safety of people, an increase in surveillance is needed. Surveillance by people is expensive and difficult to realize 24 hours a day on all places. In nowadays, surveillance systems by cameras and other sensors are used. There is a requirement for smart cameras, again to reduce the effort of human operators.

The violence do not only occur at the high risk places such as disco and pubs, but also at the public places which are supposed to be safe, e.g. train station, shopping mall, home, etc. People's safety, security and belongings are the biggest worry for most of the people nowadays. Justice does not always play an important role, when people's own safety is at stake. Recently the insurance companies are complaining that the number of stolen cars has increased a lot, despite the better and advanced locking system [4]. The main reason is that the traditional way of securing may not be effective. Nowadays people do not react, when it is clear that somebody is trying to steal a car. For their safety it is wiser to pass by and pretend that they have not seen anything. Therefore sound alarms are not as effective as in the past now. They will most probably be ignored.

1.1. Security in Home Environment

Nowadays people have to be very careful when they leave their house behind, especially when they go on holiday or to work. Insurance companies will not cover the losses when there is no strong evidence that people have attempted to hold back the burglars. They expect people to secure their house as much as possible to discourage and delay the thieves. Gross omission of the owner that causes fire damage to his house will also not be covered by the insurance. Therefore a good and reliable maintenance of their house is necessary [11]. The most common way to secure house can be divided in 4 categories: discourage, prevent, detect and register [12][13][14][15].

Discourage:

- Pretend that there is someone in your house, even there is no one present. A lively house discourages the intruders. E.g. Turn the light on when it is getting dark.

Prevent:

- Time and safety of the intruder are inversely proportional. The more time it costs to break in, the less safe the intruder feels. E.g. Use good locks for your doors and light up your balcony and garden to prevent intruders breaking into the house without being noticed.

Detect:

- Dogs. Dogs can detect intruders from miles away and they are also able to attack the intruder. But he can scare the people who just passed by.
- Detection system. When the door has been opened abnormally, the detection system can alarm the police. But the false alarms can become a nuisance as well as potentially dangerous.

Register:

- Surveillance cameras which can be monitored by a professional security company can provide the surveillance guards or police the needed information to track and identify the intruder.

1.2. Possible Security Units in Home Environment

What system can be more effective and reliable than sound alarms and static cameras when money also plays an important role? There are 3 abilities concerning a security unit: able to detect the intruder at any time, able to stop the intruder at any place and able to save the evidence (images or sounds of the intruder). The ideal security unit fulfills these requirements perfectly and is able to execute them with a success rate of 100%. One way for approaching this ideal security unit is a robot. Robots can be operational for 24 hours a day and 7 days a week. Depending on the robot abilities the 3 security abilities can be carried out by the robot. Table 1.1 shows a comparison between the 4 security units. As the table shows the robot has high potentials to replace the other 3 security units or use it as an additional security unit to cover the weak points of the other security units. But to achieve a robot which can really carry out these tasks in practice, a lot of problems still need to be conquered.

Table 1.1. Comparison table of the security units.

	Camera	Guard	Dog	Robot
Detect intruder	sometimes	often	often	usually
Stop intruder	never	often	often	possible
Save evidence	yes	no	no	yes
Costs	low	high	average	average
Development state	Matured	Matured	Matured	Start phase

1.3. Multimodal Systems

Human beings have five senses to interact with its environment: sight, hearing, smell, taste and touch [6]. These five basic senses are necessary to interact with the environment efficiently. Some animals have more than these five senses to survive in their world. According to our research there are 4 sensors which are in a more matured state in the robotic field: sight, hearing, touch and smell. The more sensors one have, the more information one will get and more appropriate actions can be determined for a certain situation. Cameras are unimodal systems that can only sense sight. Therefore image detection is the only ability of cameras.

The Sony quadruped robot dog, AIBO, is a good example which carries 3 of these 4 sensors, sound, image and touch sensors. By using these 3 sensors and its corresponding programs AIBO is able to detect abnormalities in images, detect abnormal sounds and conquer bad sight using its touch sensor. Many researchers have chosen for the AIBO as their research medium for robotic research because of its completeness and low cost [8][9]. Because of these reasons this robot will also be used as a medium for this project.

Although there are a lot of benefits by using multimodal systems compared to unimodal system, there are problems that need to be conquered. Using more sensors means more inputs. More input information can be very helpful to decide the most appropriate action in a certain situation, but these inputs will be processed in an intelligent way to produce the most suitable output, the reaction. This means that the system is able to process all inputs on time, information overload have to be avoided. This implicates that a good architecture has to be used for an effective concurrent processing of data and data fusion [10]. In order to choose the most appropriate action the inputs of all sensors have to be delivered on time before processing. Otherwise the decision is based on the wrong state. The way of concurrent processing of data and data fusion of the sensors inputs are the key issues for the success of a multimodal system.

1.4. Thesis Motivation and Challenges

Motivation:

Usually a multimodal system, e.g. robot, consists of 4 main stages: receiving input information from sensors, processing the sensors information, reasoning for appropriate reactions and executing these reactions by the output actuators. These 4 stages form a cycle and it is a continuing process that can last forever.

To create the ideal multimodal robot which can secure the home environment, it is necessary to have as many sensors as possible. More information can lead to a better understanding of the situation. An intelligent reasoning system is needed to choose the most appropriate reaction depending on the situation. Thereafter the multimodal system has a range of possible output actuators to react on its environment.

As concluded before the author has decided to use the Sony robot dog AIBO to approach this ideal robot watchdog. The Sony AIBO has 4 kinds of input sensors: image, sound, touch and distance, and 5 kinds of output actuators: wireless communication, image, sound, physical movement and led display on its face. The reasoning system that connects the input modalities and output modalities is not available. Therefore we are required to develop a reasoning architecture. Our reasoning architecture contains a detection component, reasoning component and action component. These components collaborate with each other and their outputs are optimized.

Challenges:

- Fusion of sound and image is a complex process. If we use a reasoning system which is based on sound and image, contradiction will certainly arise. The challenge is how we are going to deal with this contradiction.
- A new architecture for the AIBO watchdog has to be developed. This architecture makes sure that the detection component, reasoning component and action component work correctly and let them collaborate effectively with each other to optimize their output.
- The reasoning component produces the commands that and where AIBO needs to carry out for its security functions and it also deals with the navigation in its world environment. An intelligent algorithm needs to be developed to let AIBO navigate properly in its environment without damaging the AIBO itself.

1.5. Research Topic

The challenges described in the previous section results in the general research topic of this thesis. In general the purpose of this thesis is to create a security system for the home environment by using the AIBO. The main focus is on the reasoning component and reaction component of the security system, since the AIBO lacks a decent reasoning component. The output part is necessary to show the correct working of the reasoning component. The general research topic can be summarized as:

<u>Design an intelligent reasoning system that is able to process information from the environment and to generate appropriate reactions of the AIBO to take care of the home environment.</u>

In the sensors detection components the image sensor, sound sensor and distance sensor will be used. The reasoning system will be developed from scratch and having the ability to reason about the data from the 3 input sensors and choose the appropriate outputs. At the output side the wireless communication, image, sound and physical movements will be used to show the results of the reasoning system.

The intelligence of the AIBO watchdog is determined by the reasoning component. Having a more intelligent AIBO is necessary to operate in a more chaotic environment. In one of the most extreme situation there are a lot of objects and they can not easily be identified e.g. a half burned book, or the object does not belong to that location, e.g. a bed in the kitchen. The following reasoning component ability forms the base of this project: ability to know related events and understand the degree of importance of every event in all situations, so that the most appropriate reaction can be executed.

The topic concerning the design and implementation of the solution that has been described in this chapter can be split in some more detailed subtopics. The research assignment is therefore the following:

- Design an architecture to let the sensors detection component, reasoning component and action component collaborate efficiently with each other.
- Design an intelligent reasoning component which is capable to collaborate with the sensors detection component and action component.
- Design the world model of the AIBO watchdog and its related navigation algorithm.
- Implement a prototype which can prove the proper working of the designed architecture and reasoning component, in the designed world model.
- Test this prototype to see whether the designed system and approach perform as expected.

The developed system will be a proof of concept and it is able to present the possibilities of the AIBO watchdog. Figure 1.1 shows the focus of this AIBO watchdog project.

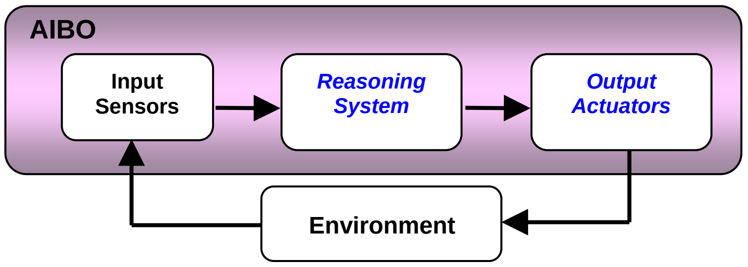


Figure 1.1: The two blue components of the AIBO watchdog show the focus of this AIBO watchdog project.

1.6. Approach

After understanding the research assignment we have to know the details about the related research on this field. A literature survey has to be conducted about the AIBO as a security system. The survey must at least contain the following topics: current available reasoning models, ways to use multimodalities and the current problems and current route planning systems.

An analysis has to be made about these topics and the useful theories and algorithms has to be listed which can be used for the AIBO watchdog. Advantages and problems when applying these theories and algorithms on the AIBO need to be researched.

The home environment where the AIBO should operate needs to be described by a world model. This world model will give the AIBO knowledge about its environment and the relationship between objects. A new design of the reasoning system has to be developed to cooperate with the world model. This design can be fine-tuned by using the theories from the literature research. Additionally a complete new architecture has to be developed to synchronize the reasoning system with the physical movements and technical outputs of the AIBO. An intelligent navigation approach of the AIBO watchdog in its environment needs also to be designed. A new route planner will be added to the architecture to increase the usefulness of the AIBO watchdog.

After acquiring the architecture design the whole architecture will be implemented and tested. By developing the test scenarios we are able to test the success of the developed architecture. As a completion of this thesis project the created security system, AIBO watchdog, will be evaluated and based on the evaluation recommendations will be made for future work.

1.7. Outline of Thesis

The thesis report consists of nine chapters. The first chapter gives an introduction about the project and the problem definition that need to be solved. Chapter 2 explains the related background knowledge about the design concepts and terms for the AIBO watchdog. Chapter 3 presents related research about AIBO, and particularly the research to design AIBO as a watchdog. Chapter 4 provides the basic knowledge of AIBO and its functionalities. Chapter 5 presents the models and concepts to design the AIBO watchdog. It encloses the detailed explanations of the design. Chapter 6 discusses the software design of AIBO watchdog based on the concepts and models introduced in Chapter 5. Chapter 7 describes the implementations of the developed architecture for AIBO watchdog. Chapter 8 discusses the test methodology and test results of the design and implementations. Conclusions and recommendations regarding the whole project are given in Chapter 9.

2

Artificial Intelligent System

"A journey of a thousand miles begins with a single step."
Ancient oriental philosophy

This chapter explains the basic knowledge of artificial intelligent systems. First an introduction will be given about artificial intelligence and the global working of an artificial intelligent system. Thereafter the aspects of the artificial intelligent system will be discussed. Many examples will be provided for each aspect of the artificial intelligent system. At the end of each aspect a conclusion will be drawn regarding the use of these aspects for the development of the AIBO watchdog.

2.1. Introduction to Artificial Intelligence

An agent is an autonomous system which is able to perceive its environment through sensors, processing the sensed information, reason for the actions based on the sensed information and acting upon that environment through actuators. This basic concept is illustrated in Figure 2.1.

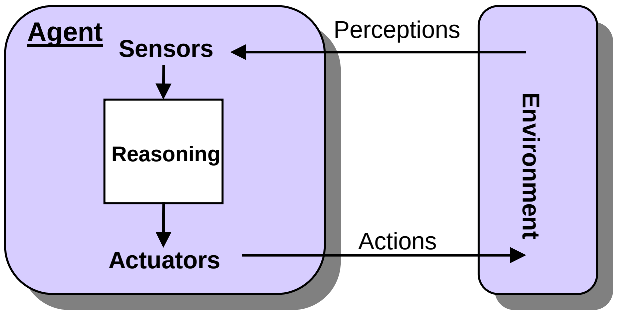


Figure 2.1. Agent's interaction process with environments through sensors and actuators.

Intelligent robots are seen as rational agents in the Artificial Intelligence world. They are making the most appropriate decisions based on the information that they have. These decisions are called the rational decisions.

The principal requirements for a robot [23]:

- Obtaining information from the outside world using visual or auditory sensors;

- Matching the information with the internal database to understand the environment;
- Designing an appropriate plan to execute a given task;
- Handling unexpected events, arriving either from the outside world or from the robot itself;
- Learning from experience to improve its performance.

A human agent has eyes, ears, and other organs as sensors and hands, legs, mouth, and other body parts as actuators. A robotic agent might have cameras and infrared range finders as sensors and wheels and legs as actuators. A general assumption can be made that every agent can perceive its own actions, but not always the effects.

The term percept is to refer to the agent's perceptual inputs at any given instant. An agent's percept sequence is the complete history of everything the agent has ever perceived. In general, an agent's choice of action at any given instant can depend on the entire percept sequence observed to date. If we can specify the agent's choice of action for every possible percept sequence, then we have said more or less everything there is to say about the agent [38].

"For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has [38]. "

There are 4 kinds of agents: simple reflex agents, model-based reflex agents, goal-based and utility based agents. Simple reflex agents respond directly to percepts, whereas model based reflex agents maintain internal state to track aspects of the world that are not evident in the current percept. Goal-based agents act to achieve their goals, and utility-based agents try to maximize their own expected "happiness" which is defined by a utility function [38].

2.2. Agent's Environment

After understanding the basic concept of the agents' interaction with its environment in this section the in depth details about the agents' environment will be explained. Depending on the complexity of the environment a reasoning agent will be chosen. The simpler environment, the simpler the reasoning system of the agent will be. The environment can be divided in 6 dimensions.

1. **Fully observable vs. partially observable:**

   In a fully observable environment an agent's sensors will have access to the complete state of the environment at each point in time. On the other hand, in a partially observable environment the agent will miss some important data about its environment due to inaccuracy of sensors or lack of certain sensors.

2. **Deterministic vs. stochastic:**

   In a deterministic environment the next state of the environment is completely determined by the current state and the action executed by the agent. Otherwise it is stochastic.

3. **Episodic vs. sequential:**

   In an episodic environment the next action does not depend on the actions taken in previous state. But in a sequential environment the current decision can affect all future decisions.

4. **Static vs. dynamic:**

   If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise it is static.

5. **Discrete vs. continuous:**

   This dimension tells about the state of the environment. In a discrete situation there are a finite number of distinct states. In a continuous-state environment, on the other hand, there is countless number of continuous states. These are caused by the time aspect or other continuous state variables.

6. **Single agent vs. multi agent:**

   Single agent works alone and there are no other agents involved in its environment. Multi agents have to collaborate with other agents or compete with them.

Depending on the results of the 6 dimensions of the agent's environment the complexity of the agent architecture will be determined. If we use these 6 dimensions to describe the environment of the AIBO watchdog, it will result to the following list:

1. partially observable
2. stochastic
3. sequential
4. dynamic
5. continuous
6. single agent

After having this list we can forecast the problems which the reasoning system should solve. Since the environment is partially observable the agent needs to maintain the internal states to keep track of the world. Uncertainty of the stochastic environment can cause difficulties to make a rational decision. The sequential environment forces the AIBO watchdog to think ahead and make the correct decision every step. In a dynamic environment the AIBO needs to keep looking at the world to perceive sudden changes. As result the reasoning system should be a goal-based agent.

These are the problems that we have to deal with, when designing the reasoning system of the AIBO watchdog. Therefore new concepts have to be developed to solve these problems or work around them.

2.3. Agent Architectures

The agent architecture can predict and explain the behavior of an agent system based on its current state and that of the environment. Furthermore it also provides a methodology or blueprint of how to go about building the real agent system. Therefore we will provide a general background to agent systems by showing some of the variety of ways in which the agent metaphor has been interpreted and implemented. Benefits and limitations of each approach will also be discussed. There are 3 agent architecture categories for single agents: reactive architectures, deliberative architectures and hybrid architectures [28].

- *Reactive agent systems* act by means of stimulus-response rules and do not symbolically represent their environment.
- *Deliberate agent systems* symbolically model their environment and manipulate these symbols in order to act.
- *Hybrid agent systems* can act both deliberatively and reactively.

In general, architectures provide information about essential data structures, relationships between these data structures, the processes or functions that operate on these data structures, and the operation or execution cycle of an agent [27].

2.3.1. Reactive Agent Architecture

The traditional AI view is that in order for software agents to exhibit intelligent behavior, they need an internal (symbolic) representation of their environment. The agents can manipulate and reason about this internal representation in order to determine what to do next. However, some have argues that effective behavior does not necessarily require symbolic representations and manipulation, a view strengthened by the problems that have emerged in mainstream artificial intelligence such as the complexity and, in some cases, intractability, of some symbolic manipulation problems such as planning. This is because symbolic reasoning is very resource and time-intensive in determining the best action to perform next, so that by the time the action is performed, the environment may have changed in such a way that the action is no longer useful. In this case, the agent could be said to have failed. The opposite view is that effective behavior is only achieved when systems are situated and embodied in the real world, and can respond to events in the environment in a timely fashion. In this view, agent behavior is directly coupled with the world, typically incorporating stimulus-response rules; the environment provides a stimulus that causes a rule to fire and the agent to respond in specified way.

Agents that do not maintain a symbolic representation of their environment are known as reactive agents, and their architectures as reactive architectures. Such reactive systems were originally proposed by Brooks, who developed the subsumption architecture for controlling the behavior of a robot [27].

Subsumption Architecture

Brooks proposed the subsumption architecture as a means of controlling the behavior of a mobile robot in real time, with three basic requirements [29], as follows:

- Agents should be able to cope with multiple goals. For example, an agent might have the goal of moving to a certain location in its environment, directly ahead of it, as quickly as possible, while at the same time also possessing the goal to avoid obstacles. Clearly, when there is an obstacle in the path of an agent, these goals conflict, and any control mechanism for the robot needs to determine which of the goals takes priority.

- Agents should have multiple sensors. In general, physical agents have several different sensors, each with the possibility of giving erroneous or conflicting readings. Agents must be able to make decisions in these difficult circumstances.
- Agents should be robust. Essentially, this refers to two aspects: first, agents must be able to function (albeit possibly less effectively) when some sensors fail; second, agents should be able to behave effectively when the environment is changing drastically around them.

The approach taken by Brooks is not a functional one in which component parts are isolated according to perception, action, reasoning, planning and so on, with some overarching central control mechanism. Instead, it is achieved using an entirely different approach, the layered approach. Here, the architecture comprises several task-achieving behaviors, each of which is implemented separately and arranged as shown in Figure 2.2. This hierarchy of layers reflects how specific a behavior is; the more specific the task, the higher the layer. In the case of the mobile robot, there are eight layers of behavior, from 1 to 8, as follows [27]:

1. Avoid contact with objects in the environment;
2. Wander randomly;
3. Explore the environment;
4. Build a map of the environment and plan routes within it;
5. Notice change in the environment
6. Reason about and identify objects in the world, and perform certain tasks related to certain objects;
7. Build and execute plans that involve changing the world in some desirable way;
8. Reason about the behavior of other objects and how it might impact on plans, and consequently modify plans.

The different layers exist in parallel, with each unaware of the other layers that are above it, but able to examine data from lower layers. Each layer is connected to the sensors and actuators, but extracting only those aspects of the world (from the sensors) that is relevant to its function. In addition, layers can prevent lower layers from trying to control the behavior of the agents by suppressing their inputs and inhibiting their outputs.

The first step in the agent's construction is to build the 0th control layer and, once this has been tested, to build the 1st control layer on top of the 0th layer. The 1st layer has access to the data at layer 0 and can also inject data to suppress the normal activity of the 0th layer. The 0th layer continues to execute, unaware that there is a higher layer intermittently influencing its behavior. The process can then be repeated for each successive layer, with each layer competing to control the behavior of the robot. In this way, the control system functions at a very early stage of system development, with higher layers being added as required without having to change the existing lower-layer architecture.

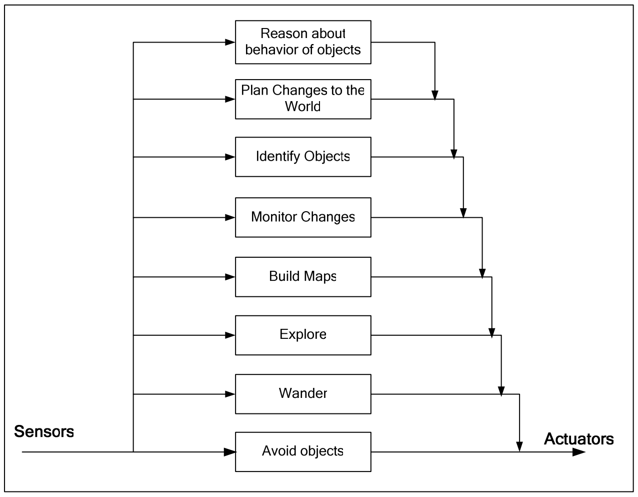


Figure 2.2: The subsumption architecture which consists of 8 layers represents an example of the reactive agent architecture.

2.3.2. Deliberative Agent Architectures

Agent systems that are able to maintain and manipulate representations of the world, without stimulus-response rules are called deliberative agents [28]. In order to model rational or intentional agency in these kinds of agents, mentalistic notions, or mental attitudes are used to describe and characterize behavior. These attitudes include beliefs, goals, desires, knowledge, plans, motivations, and intentions, and are commonly grouped into three categories: informative, motivational, and deliberative [31]. The first category refers to aspects that a system considers to be true about the world, and includes knowledge, beliefs and assumptions; the second refers to the wants of a system, including goals, desires, and motivations; and the third concerns how an agent's behavior is directed and includes plans and intentions. The distinction between the second and third categories is subtle, since it is possible that a system may desire a certain state without planning for it, or without intending it to happen.

There are several compelling reasons why agents defined using mental attitudes might be useful. First, if an agent can be described in terms of what it knows, what it wants, and what it intends, then, since it is modeled on familiar concepts, it becomes possible for users to understand and predict its behavior. Second, understanding the relationship between these different attitudes and how they affect behavior can provide the control mechanism for intelligent action in general. Third, computational agents designed in this way may be able to interpret the behavior of others independently from any implementation.

Many agent systems include a deliberative architecture to support reasoning at the mental-attitude level. Moreover, many of these deliberative architectures are based on the belief-desire-intention (BDI) model of rational agency [27].

BDI architecture

The BDI model provides a foundation for many systems. Architectures based on the BDI model explicitly represent beliefs, desires, and intentions as data structures, which determine the operations of the agent. The intuition with BDI systems is that an agent will not, in general, be able to achieve all its desires, even if these desires are consistent. Agents must therefore fix upon some subsets of available desires and commit resources to achieving them. These chosen desires are intentions, and an agent will typically continue to try to achieve an intention until either it believes the intention is satisfied, or it believes the intention is no longer achievable [32]. Figure 2.3 shows the global working of a reasoning system, PRS (procedural reasoning system), which is based on the BDI architecture.

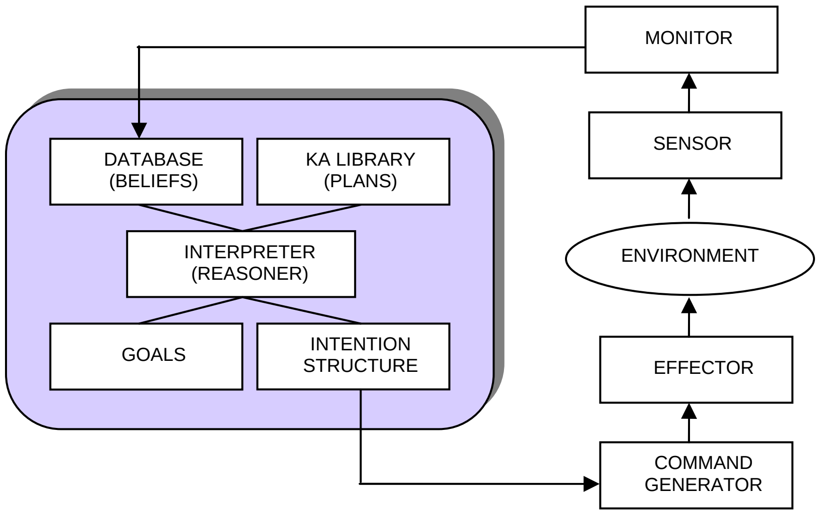


Figure 2.3: A representation of PRS architecture with its internal components.

At the start of execution, agents are initialized with a set of plans, goals, beliefs, an empty event buffer and no intentions. The operation of the agent can then be enumerated as follows:

1. Perceive the world, and update the set of events.
2. For each event, generate the set of plans whose trigger condition matches the vent. These are known as the relevant plans of an event.
3. For each event, select the subset of relevant plans whose context condition is satisfied by the agent’s current beliefs. These plans are known as active plans.
4. From the set of active plans, select one for execution so that it is now an intention.
5. Include this new intention in the current intention structure either by creating a new intention stack or by placing it on the top of an existing stack.
6. Select an intention stack, take the topmost intention, and execute the next formula in it.

2.3.3. Hybrid Agent Architectures

In general, agents can be neither totally deliberative nor totally reactive. If they are only reactive, they cannot reason about their actions and will not be able to achieve any sophisticated behavior; if they are just deliberative they may never be able to act in time. It is generally recognized that if agent systems are to survive in real and complex environments they need to be reactive in order to respond to environmental changes with sufficient speed, and be deliberative in order to achieve complex goals without deleteriously affecting longer-tem option [52]. If environments change rapidly or unexpectedly, agents may need to act in a reactive manner, whereas more stable environments may allow agents time to deliberate on the best course of action. Architectures containing both deliberative and reactive components are hybrid architectures. Figure 2.4 shows an example of a hybrid architecture, TouringMachines [30].

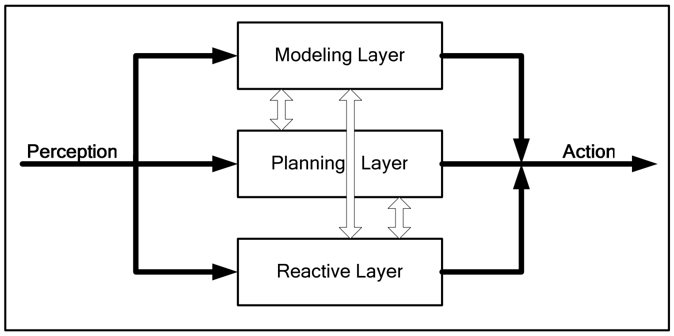


Figure 2.4: Representation of the architecture of TouringMachines, which is an example of the hybrid architecture.

This system was designed with the following issues in mind:

- They need the ability to deal with unexpected events in the real (physical or electronic) world, and do so at different levels of granularity.
- They need to deal with the dynamism in the environment created by the actions of other agents.
- They must pay attention to environmental change.
- They need to reason about temporal constraints in the knowledge that computation is necessarily resource-bounded.
- They must reason about the impact the short-term actions may have on long-term goals.

This architecture is similar to the subsumption architecture in that it consists of a number of layers, the reactive layer, the planning layer, and the modeling layer, which continually compete to control the agent’s behavior. The reactive layer responds quickly to events not explicitly programmed in the other layers, such as when a new agent or obstacle is perceived. Generating, executing and modifying plans, such as constructing a route in order to move to a target destination, are the responsibilities of the planning layer. Finally, the modeling layers is used for building and maintaining models of entities in the environment, which are used to understand the current behaviors of others and make predictions about their future behaviors.

2.4. Knowledge Representation

For the answer of the question, “Why do people in AI who want their systems to know a lot, also want their systems to represent that knowledge symbolically?”, there is a clear reason. Much of the work in AI wants to construct systems that do contain symbolic representations with two important properties. First is that we can understand them as standing for propositions. Second is that the system is designed to behave the way that it does because of these symbolic representations [41]. Since the start of Artificial Intelligence in the 1950’s several knowledge representation formalisms have been developed. These knowledge representations describe the way that the data knowledge will be stored. The most well known will be briefly discussed here.

2.4.1. Production Rules

One of the most common knowledge formalisms used in AI is the standard production rule. Their popularity is partly due to the fact that they have turned out to be very useful in the construction of expert systems. A production rule consists of a condition-action pair. If the condition part of the rule is met, the rule is fired and the action in the action part of the rule is carried out. The format of a rule is:

IF A THEN B

Where A is the condition part and B is the action part. A can consist of a number of premises, A1, A2, …, An, joined by various connectives. The IF part is also known as the antecedent of the rule, while the THEN part is the consequent of the rule. For example, if a AIBO ball is known to have a pink color and also a round shape, then the following rule can represent this:

IF Pink AND Round THEN AIBO Ball.

Many expert systems use rules to represent their information [33]. There are two types of rules: conjunctive and disjunctive. A conjunctive rule is defined as a rule containing attributes linked by the AND connective. Therefore, the previous rule is a conjunctive rule. This is the original form of production rules utilized in knowledge based systems. However, most knowledge-based system building tools today also permit the use of other connectives. A disjunctive rule is represented as attributes (which can be in the form of a conjunctive rule) linked together by the OR connective. For instance:

IF (Pink AND Plastic AND Hard) OR (Pink AND Round) THEN AIBO Ball

is a disjunctive rule. This rule is equivalent to the following 2 conjunctive rules:

IF Pink AND Plastic AND Hard THEN AIBO Ball.

IF Pink AND Round THEN AIBO Ball.

Conjunctive rules correspond to a mapping between the input space and the output space, as shown in Figure 2.5. Distinctive rules, on the other hand, correspond to mapping such as those in Figure 2.5 for a rule with one OR connective.

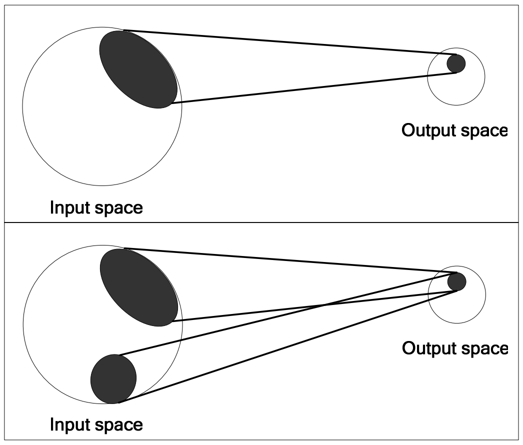


Figure 2.5: Explanation of conjunctive and distinctive rules. The figure on the top shows the conjunctive rules mapping between input space and output space. The figure on the bottom shows the disjunctive rules mapping with one OR connective between input space and output space.

Advantages

- Naturalness of expression. It is claimed that production rules often provide the right grain size at which to represent the knowledge that expert problem solvers use. This makes it relatively straightforward for the domain experts to understand the rules [34].
- Modularity. Permanent knowledge is separated from temporary knowledge. Production rule systems contain both a rule base, in which the more permanent knowledge resides, and a working memory, which contains the temporary knowledge describing the problem the system is currently working on.

  Different rules are structurally independent. Each production rule represents a specific piece of knowledge that is completely independent of the other rules. Therefore maintaining and constructing a rule base is easier.

  The fact that the rules are independent of each other facilitates the incremental construction of a knowledge base.
- Simple problem solving method. Production rules determine what to do next by examining the representation of the present state of the problem solving process in working memory. First, minimal changes to working memory can quickly lead to important focus shifts in the problem-solving process. Second, because the whole state of working memory is open to inspection, production rule systems can pursue a large number of different solutions at any time.

Disadvantages

- Efficiency problems. With large rule bases determining the conflict set may be an expensive process. If the conflict set is very large, the conflict resolution may become very expensive.
- Restricted syntax and expressibility problems. Certain types of knowledge can not be expressed easily or knowledge that can not be expressed at all. The inability to express

structural knowledge leads to these strange looking rules, with the problems that this entails for intelligibility and maintainability of large rule bases [34].

2.4.2. Decision Trees

A decision tree is a directed graph showing the various possible sequences of questions, answers, and classifications. The decision tree may consist of a test that has a set of mutually exclusive possible outcomes together with a subsidiary decision tree for each such outcome. The subsidiary decision tree may be a test or, alternatively, it may be a leaf. The process of classifying an object starts at the root of the tree. If this is a leaf, the object is assigned the class associated with that leaf. Alternatively, if the root of the decision tree is a test, the outcome of this test is determined for the given object and the process continues using the subsidiary decision tree of that outcome. An object is thus classified by racing out a path from the root of the decisions tree to one of its leaves. Figure 2.6 shows a simple decision tree.

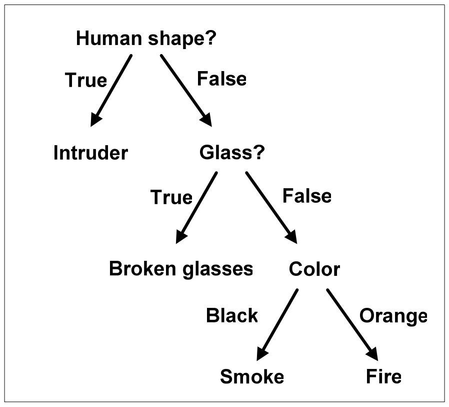


Figure 2.6: Simple decision tree which can determine the object by answering the questions at the roots.

Advantages

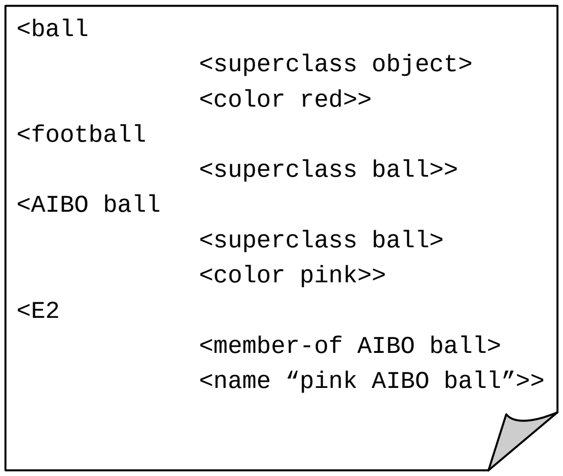
- Decision trees can have discrete or continuous values as its input and discrete or continuous values as its output.
- Decision tree representation is very natural for humans and many "How to" manuals are explained in decision trees [38].
- Decision trees are fully expressive within the class of propositional languages. Any Boolean function can be written as a decision tree.

Disadvantages

- Decision trees are not capable to represent tests that refer to two or more different objects.
- Decision trees grow exponentially when certain functions are used, the parity functions and majority functions. Parity functions returns 1 if and only if an even number of inputs are 1. Majority functions returns 1 if more than half of its inputs are 1 [38].

2.4.3. Frames

Frames are defined as data structures for the representation of stereotypical situations [55]. Frames are retrieved whenever the system encounters a new situation. They are formed on the basis of previous experiences in similar situations and can best be seen as a structure representing expectations of the system about situations of this kind. Information is stored in a frame by associating descriptions with it. Thus, they are descriptions of objects. The descriptions in a frame are called slots. A slot usually consists of two parts: a slot-name, which describes an attribute, and a slot-filler, which describes either a value for that attribute or a restriction on the range of possible values. Figure 2.7 illustrate a knowledge base based on frames.



```
<ball
             <superclass object>
             <color red>>
<football
             <superclass ball>>
<AIBO ball
             <superclass ball>
             <color pink>>
<E2
             <member-of AIBO ball>
             <name “pink AIBO ball”>>
```

Figure 2.7: An example content from a frame-based knowledge base.

As Figure 2.7 illustrates frame-based knowledge representations have a hierarchical nature. The way to use this knowledge consists of a few steps. The first step to use this information in the reasoning process concerns the discovery of those frames that can be applied to the situation the system currently finds itself in. Given a partial description of the situation, find the frames in the knowledge base that are consistent with this situation. Thus, the system has to match a description of the specific situation that it is facing with a general description of situations of this kind.

Advantages

The advantages of the frame-based knowledge representation languages are mostly at the epistemological level and concern the way in which knowledge can be organized in frame-based representation languages.

- Frame-based knowledge representation languages capture the way in which domain experts typically think about their knowledge. The structure of the domain about which the knowledge is modeled is directly reflected in the knowledge base [42].
- The way in which entities are described. The most important way of describing an entity in frame-based knowledge representation languages is by specialization, that is, by comparing the entity to other things that you already know about.

Disadvantages

The disadvantages of frame-based knowledge representations language are mostly at the logical level and concern such things as the meaning and the expressive power of the formalism.

- Absence of a clear semantics for frame-based representation languages. None of the frame-based representation languages that were originally proposed had a semantics defined for them.
- Problems from the use of default heritance for inference. The question is what exactly is not being inherited when a class frame or instance frame does not inherit a slot associated with a higher class frame. We can make a distinction between the value of slot not being inherited, and the slot itself not being inherited.
- Problems with the expressive power of frames. Expressing existential knowledge is very difficult to realize [34].

2.4.4. Semantic Nets

However semantic nets are very similar to frames, there is a clear difference between them. Semantic network representations are primarily based on the interconnectivity intuition and frame-based representations stress the intuition that knowledge should be organized in larger chunks. The basic idea of a semantic network representation consists of two types of primitives: nodes and links or arcs. Links are unidirectional connections between nodes. Nodes correspond to objects, or classes of objects, in the world, whereas links correspond to relationships between these objects. Nodes and links are often labeled using a mnemonic device so that the user of the network language knows their intended meaning. The important point is that there is no information stored at a node. All the knowledge is represented by the links between the different nodes. Thus, the information stored in a node is simply the set of links that impinge on it. Figure 2.8 illustrates an example of this idea.

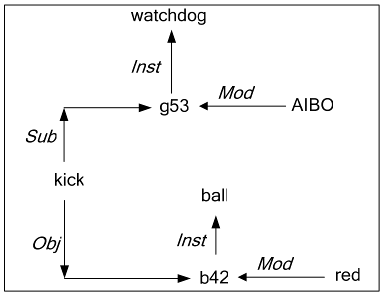


Figure 2.8: Semantic nets representation of the sentence: The AIBO watchdog kicks the red ball.

This kind of representation rather defines the meaning of a sentence than the meaning of a word. The meanings of the annotations Mod (Modifications), Sub (Subject), and Obj(Object), are self evident. The Inst (Instance of) link is used as a link between a token node and its type node. Thus g53 and b42 are token nodes which have Inst links to the types nodes watchdog and ball. G53 and b42 represent the particular watchdog and the particular ball mentioned in the sentence [33][34].

Advantages

- Fewer explicit inference rules are needed.
- Most of the inferences that would otherwise have to be drawn are already implicit in the representation. Thus, if everything was stored in terms of a few primitives, then synonymous sentences would be stored in exactly the same format.

Disadvantages

- A lot of work needs to be done to translate a sentence into the underlying representation.
- It is often either impossible or very difficult to find the right set of primitives. An example of a domain where it seems difficult to give a set of primitives is that of kinship relationships [43].
- There is still a lot of inferential work that needs to be done, even if information has been stored in terms of primitives.
- Primitive representations are often rather complex and may require a lot of storage.

2.4.5. Conclusion

After reviewing the 4 most popular knowledge representation methods we need to decide the knowledge representation method for application in this AIBO watchdog project. Each method will be discussed briefly below.

Semantic web

As we mentioned already semantic web is a representation of knowledge comparable to human brain (and dog brain). So in principle it is possible to use semantic web as knowledge representation. The disadvantage of semantic web is that it is necessary to link objects with different strength. Usually the strength between objects is obtained by learning methods. So after we have a running prototype of our watchdog it is possible that we use semantic web for adaptive learning.

Frames

Frames are very useful to group objects which belong to each other. This method can be very helpful to develop the reasoning system of the AIBO watchdog.

Decision trees

A decision tree is a method that looks like production rules. The classification of objects by decision trees can also be achieved with if-then rules, but decision trees can also operate with probabilities. The probabilistic version of decision trees is the basis for the reasoning method of Bayesian networks. This method will be discussed later in this chapter.

Production rules

Production rules is one of the most used knowledge based system in application. It is very natural for human beings to represent knowledge as if-then rules. To create a reasonable reasoning system in a relatively short period this is one of the first choices [54].

Frames, production rules and decision trees are the potential candidates for this AIBO watchdog thesis project. The choice of the knowledge representation method is also dependent on the kind of reasoning system that the AIBO watchdog uses. Not all knowledge representation method forms the best combination with the reasoning system. Moreover about the reasoning systems will be discussed later in this chapter.

2.5. Meaning of Knowledge

After knowing the ways to represents the knowledge we need to know why we wants to have knowledge and what exactly are knowledge. We understand that knowledge can help the human beings make rational decisions, but how can we achieve knowledge. In the data mining world, automated extraction of novel and interesting information from large data sets, they are already trying to extract knowledge from raw computer data. Their process is similar to the concept of Professor Russel Ackoff [39]. According to Professor Russel Ackoff the content of the human brain can be classified into 5 categories:

1. Data: symbols.
2. Information: data that are processed to be useful; provides answers to "who", "what", "where", and "when" questions.
3. Knowledge: application of data and information; answers "how" questions.
4. Understanding: appreciation of "why".
5. Wisdom: evaluated understanding.

Ackoff indicates that the first four categories relate to the past; they deal with what has been or what is known. Only the fifth category, wisdom, deals with the future because it incorporates vision and design. With wisdom, people can create the future rather than just grasp the present and past. But achieving wisdom isn't easy; people must move successively through the other categories [39]. Figure 2.9: Wisdom pyramid shows the visual representation of this concept.

Data

Data is raw. It simply exists and has no significance beyond its existence. It can exist in any form, usable or not. It does not have meaning of itself, e.g. a spreadsheet generally starts out by holding data.

Information

Information is data that has been given meaning by way of relational connection. This meaning can be useful, but does not have to be, e.g. a relational database makes information from the data stored within it.

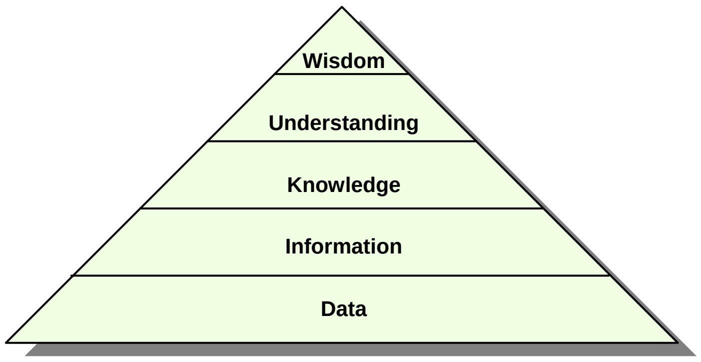


Figure 2.9: Wisdom pyramid.

Knowledge

Knowledge is the appropriate collection of information, such that its intent is to be useful. Knowledge is a deterministic process. When someone memorizes information, then they have amassed knowledge. This knowledge has useful meaning to them, but it does not provide inference to further knowledge. For example, elementary school children memorize, or amass knowledge of, the "times table". They can tell you that "2 x 2 = 4" because they have amassed that knowledge (it being included in the times table). But when asked what is "1267 x 300", they can not respond correctly because that entry is not in their times table. To correctly answer such a question requires a true cognitive and analytical ability that is only encompassed in the next level, understanding. Most of the applications we use (modeling, simulation, etc.) exercise some type of stored knowledge.

Understanding

Understanding is an interpolative and probabilistic process. It is cognitive and analytical. It is the process by which human beings can take knowledge and synthesize new knowledge from the previously held knowledge. The difference between understanding and knowledge is the difference between learning and memorizing. People who have understanding can undertake useful actions, because they can synthesize new knowledge, or in some cases, at least new information, from what is previously known and understood. That is, understanding can build upon currently held information, knowledge and understanding itself. AI systems possess understanding in the sense that they are able to synthesize new knowledge from previously stored information and knowledge [39].

Wisdom

Wisdom is an extrapolative and non-deterministic, non-probabilistic process. It calls upon all the previous levels of consciousness, and specifically upon special types of human programming (moral, ethical codes, etc.). It beckons to give us understanding about which there has previously been no understanding, and in doing so, goes far beyond understanding itself. It is the essence of philosophical probing. Unlike the previous four levels, it asks questions to which there is no (easily-achievable) answer, and in some cases, to which there can be no humanly-known answer period. Wisdom is therefore, the process by which we also discern, or judge, between right and wrong, good and bad [39]. The relationships between these 5 stages are visually presented in Figure 2.10.

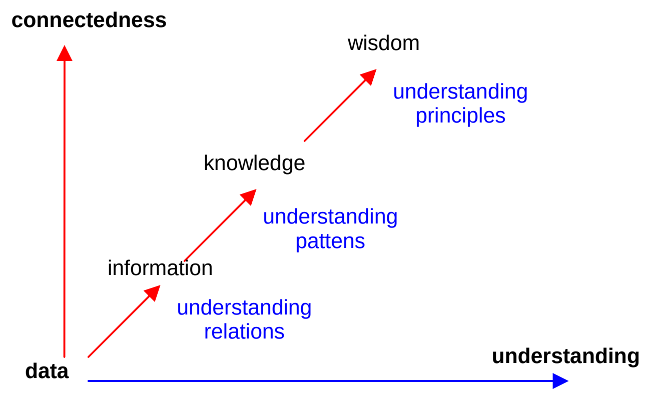


Figure 2.10: Relationships between the process stages of acquiring wisdom.

2.6. Reasoning of Artificial Agents

Reasoning is the formal manipulation of the symbols representing a collection of believed propositions to produce representations of new ones. It is here that we use the fact that symbols are more accessible than the propositions they represent: they must be concrete enough that we can manipulate them (move them around, take them apart, copy them, and string them together) in such a way as to construct representations of new propositions [35]. In this section we will briefly discuss some reasoning systems. Not all aspects and abilities of these systems will be discussed.

2.6.1. Expert System

Expert system, also known as knowledge based system, is a computer program that represents and reasons with knowledge of some specialist subject with a view to solving problems or giving advice. Such a system may completely fulfill a function that normally requires human expertise, or it may play the role of an assistant to a human decision maker. The following characteristics of an expert system can be mentioned:

- It simulates human reasoning about a problem domain, rather than simulating the domain itself. This distinguishes expert systems from more familiar programs that involve mathematical modeling. This is not to say that the program is a faithful psychological model of the expert, merely that the focus is upon emulating an expert’s problems-solving abilities; that is, performing the relevant tasks as well as, or better than the expert.

- It performs reasoning over representation of human knowledge, in addition to doing numerical calculations or data retrieval. The knowledge in the program is normally expressed in some special-purpose language and kept separate from the code that performs the reasoning. These distinct program modules are referred to as the knowledge base and the inference engine, respectively.
- It solves problems by heuristic or approximate methods which, unlike algorithmic solutions, are not guaranteed to succeed. A heuristic is essentially a rule of thumb which encodes a piece of knowledge about how to solve problems in some domain. Such methods are approximate in the sense that they do not require perfect data and the solution derived by the system may be proposed with varying degree of certainty [36].

Inference rule

An understanding of the inference rule concept is important to understand expert systems. An inference rule is a statement that has two parts, an if-clause and a then-clause. This rule is what gives expert systems the ability to find solutions to diagnostic and prescriptive problems. An example of an inference rule is:

IF the object color is pink and the object shape is a circle,

THEN the object is an AIBO ball.

As this example also shows inference rules, which are the same as production rules, are the knowledge representation of the expert system. An expert system's rule base is made up of many such inference rules. They are entered as separate rules and it is the inference engine that uses them together to draw conclusions. Because each rule is a unit, rules may be deleted or added without affecting other rules (though it should affect which conclusions are reached). One advantage of inference rules over traditional programming is that inference rules use reasoning which more closely resemble human reasoning.

Thus, when a conclusion is drawn, it is possible to understand how this conclusion was reached. Furthermore, because the expert system uses knowledge in a form similar to the expert, it may be easier to retrieve this information from the expert.

Architecture of expert systems

Expert system is also called rule-based system. It consists of inference engine, rule base and working memory. The inference engine, in turn, consists of pattern matcher, agenda and execution engine. Figure 2.11 shows the architecture of an expert system.

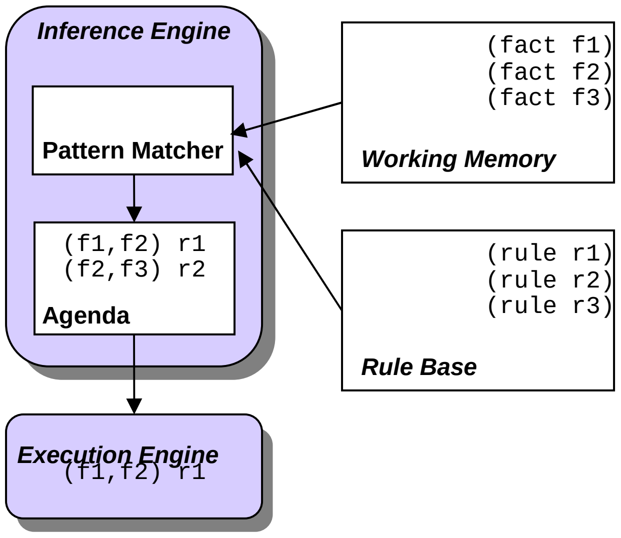


Figure 2.11: The architecture of a expert system. The pattern matcher applies the rules in the rule-base to the facts in working memory to construct the agenda. The execution engine fires the rules from the agenda, which changes the contents of working memory and restarts the cycle.

The inference engine

The primary business of a rule engine is to apply rules to data. That makes the inference engine the central part of a rule engine. The inference engine controls the whole process of applying the rules to the working memory to obtain the outputs of the system. Usually an inference engine works in discrete cycles as follows:

- All the rules are compared to working memory (using the pattern matcher) to decide which ones should be activated during this cycle. This unordered list of activated rules, together with any other rules activated in previous cycles, is called the conflict set.
- The conflict set is ordered to form the agenda—the list of rules whose right-hand sides will be executed, or fired. The process of ordering the agenda is called conflict resolution. The conflict resolution strategy for a given rule engine will depend on many factors, only some of which will be under the programmer's control.
- To complete the cycle, the first rule on the agenda is fired (possibly changing the working memory) and the entire process is repeated. This repetition implicates a large amount of redundant work, but many rule engines use sophisticated techniques to avoid most or all of the redundancy.

  In particular, results from the pattern matcher and from the agenda's conflict resolver can be preserved across cycles, so that only the essential, new work needs to be done.

The rule engine will decide the order in which the rules will be fired and this is one of the great strengths of rule-based programming. The rule engine can more or less create a custom program for each situation that arises, smoothly handling combinations of inputs the programmer might not have imagined.

The rule base

The rule engine will obviously need some location to store rules. The rule base contains all the rules the system knows. They may simply be stored as strings of text, but most often a rule compiler processes them into some form that the inference engine can work with more efficiently.

In addition, the rule compiler may add to or rearrange the premises or conclusions of a rule, either to make it more efficient or to clarify its meaning for automatic execution. Depending on the particular rule engine, these changes may be invisible to the programmer.

Some rule engines allow the designers to store the rule base in an external relational database, and others have an integrated rule base. Storing rules in a relational database allows us to select rules to be included in a system based on criteria like date, time, and user access rights.

The working memory

It is essential to store the data the 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. Typically, the rule engine maintains one or more indexes, similar to those used in relational databases, to make searching the working memory a very fast operation.

It is up to the designer of the rule engine to decide what kinds of things can be stored in working memory. Some working memories can hold only objects of a specific type, and others can include, for example, Java objects.

The pattern matcher

The 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. In general, this is a hard problem. If the working memory contains thousands of facts, and each rule has two or three premises, the pattern matcher might need to search through millions of combinations of facts to find those combinations that satisfy rules. Fortunately, a lot of research has been done in this area, and very efficient ways of approaching the problem have been found. Still, for most rule-based programs, pattern matching is the most expensive part of the process.

The agenda

Once the 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. Again, this is potentially a hard problem, and each rule engine has its own approach. Commonly, the conflict strategy might take into account the specificity or complexity of each rule and the relative age of the premises in the working memory. Rules may also have specific priorities attached to them, so that certain rules are more important and always fire first. As an example, the AIBO navigation system might have two rules like these:

IF next position is north THEN walk northwards END

IF north position contains an obstacle THEN recalculate next position END

If the next position that the AIBO needs to go is the north position, then both rules will apply. It is important that the second rule fire before the first one. This second rule should therefore be given a very high priority.

Execution engine

Finally, once the 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. In a classical production system, rules could do nothing but add, remove, and modify facts in the working memory. In modern rule engines, firing a rule can have a wide range of effects. Some modern rule engines, e.g. Jess, offer a complete programming language you can use to define what happens when a given rule fires. The execution engine then represents the environment in which this programming language executes. For some systems, the execution engine is a language interpreter; for others, it is a dispatcher that invokes compiled code.

2.6.2. Case Based Reasoning

Case based reasoning (CBR) is a methodology for solving problems by utilizing previous experience. It involves retaining a memory of previous problems and their solution and, by referencing these, solving new problems. Generally, a case based reasoner will be presented with a problem. It may be presented by either a user or another program or system. The case based reasoner then searches its memory of past cases (the case base) and attempt to find a case that has the same problems specifications as the current cases. If the reasoner can not find an identical case in its case base, it will attempt to find the case or cases in the case base that most closely match the current query case.

In the situation where a previous identical case is retrieved, assuming its solution was successful, it can be returned as the current problem’s solution. In the more likely case that the retrieved case is not identical to the current case, an adaptation phase occurs. In adaptation, the differences between the current case and the retrieved case must first be identified and then the solution associated with the retrieved case modified, taking into account these differences. The solution returned in response to the current problem specification may then be tried in the appropriate domain setting [37].

The structure of a case-based reasoning system therefore is usually devised in a manner that reflects these separate stages. At the highest level a case-based reasoning (CBR) system can be thought of as a black box (see Figure 2.12) that incorporates the reasoning mechanism and the external facets:

- The input specification (or problem case).
- The output suggested solution.
- The memory of past cases that are referenced by the reasoning mechanism.

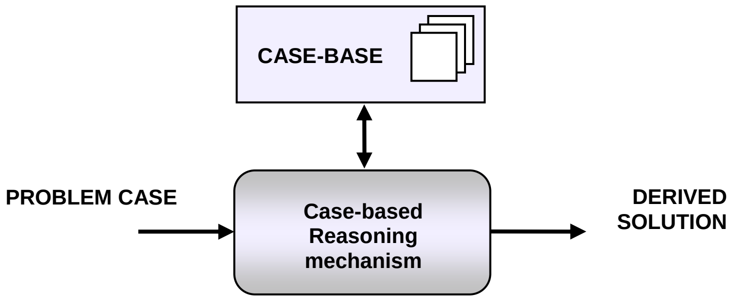


Figure 2.12: An example architecture of a CBR system**.**

In most CBR systems, the CBR mechanism can also be divided into two parts: the case retriever and the case reasoner, illustrated in Figure 2.13.

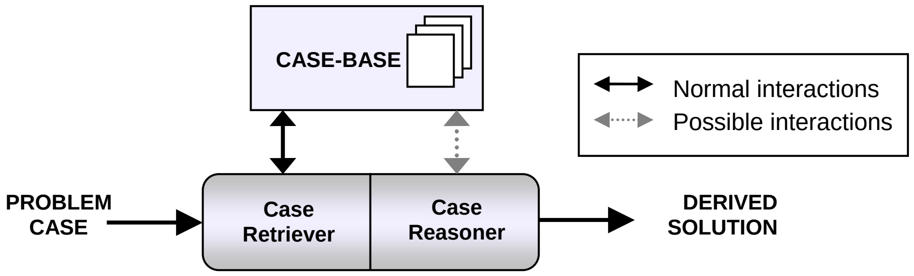


Figure 2.13: Architecture of CBR system showing the 2 major components.

The case retriever's task is to find the appropriate cases in the case base while the case reasoner uses the retrieved cases to find a solution to the given problem description. This reasoning generally involves both determining the differences between the retrieved cases and the current query case; and modifying the retrieved solution appropriately, reflecting these differences. This reasoning part itself may or may not retrieve further cases or portions of cases from the case base.

2.6.3. Bayesian Network

A Bayesian network is a directed graph in which each node is annotated with quantitative probability information. The specification is as follows:

- A set of random variables makes up the nodes of the network. Variables may be discrete or continuous.
- A set of directed links or arrows connects pairs of nodes. If there is an arrow form node X to node Y, X is said to be a parent of Y.
- Each node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node.
- The graph has no directed cycles.

The topology of the network, the set of nodes and links, specifies the conditional independence relationships that hold in the domain, in a way that will be made precise shortly. The intuitive meaning of an arrow in a properly constructed network is usually that X has a direct influence on Y. It is usually easy for a domain expert to decide what direct influences exist in the domain than actually specify in the probabilities themselves. Once the topology of the Bayesian network is laid out, we need only specify a conditional probability distribution for each variable, given its parents. We will see that the combination of the topology and the conditional distributions suffices to specify the full joint distribution for all the variables [38]. This concept is illustrated in Figure 2.14. The topology of Figure 2.14 shows that fire directly affect the probability of the smoke is present and whether the fire scenario or not depends on the fire and smoke. The conditional probability tables of smoke shows the probability of the values of smoke based on the fire value. For example there is a chance of 40% that there is smoke present when there is fire and when there is no fire that probability will be decreased till 1%.

If we consider the conditional probability table of fire scenario, we can notice that there is 1% chance that there is no fire scenario, when fire and smoke are detected.

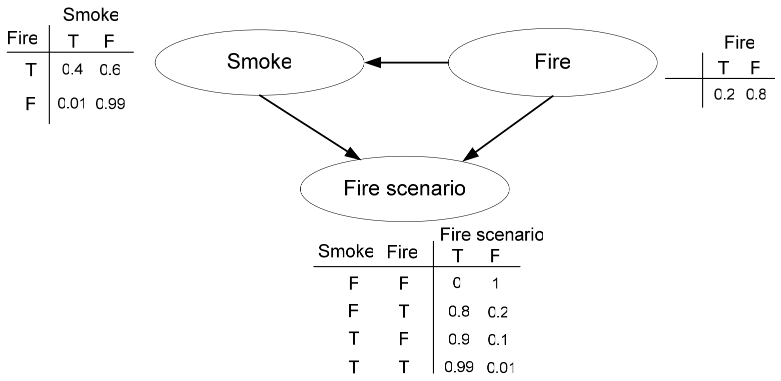


Figure 2.14: A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs).

2.6.4. Neural Networks

Neural networks were inspired by the architecture of the human brain and various mathematical theories. They comprise a set of nodes connected by weighted links and are trained by examples. Rather than being programmed, neural networks are trained with examples and acquire knowledge through experience. The advantages of this system are [44]:

- The designer does not have to worry about defining formal methods for the representation of the knowledge;
- There is no need for exhaustive searches to retrieve the knowledge.

Neural networks have two further aspects [45] that make them invaluable in real-world applications and are similar to the way humans use information.

- Generalization ability. The real world rarely presents information with the precision required. Neural networks accomplish the generalization needed by their structure rather than by elaborate programming. This is crucial in real world problems where one cannot brain a network in advance for every circumstance it might encounter in the field [46].
- Abstraction ability. Neural networks can abstract the ideal from a non-ideal training set. Through training, neural networks form an internal representation of the salient features of the training set [45].

Neural networks are built from simple units, called neurons. These neurons are connected by weighted links. During training the learning ability of the neural network will be activated. Training is by given the training examples with its related inputs and outputs, the weight of the links will be adapted using a backpropagation algorithm. After the adaptations the neural network can classify the inputs with its network and an appropriate output will be given. Figure 2-15 illustrates an example of a simple neural network.

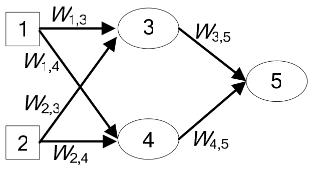


Figure 2-15: A simple neural network with two inputs, one hidden layer of two units and one output.

2.6.5. Conclusion

After reviewing the 4 most popular reasoning systems a decision needs to be made regarding the reasoning system of the AIBO watchdog. Each reasoning system will be discussed briefly below.

Expert system

Expert systems are the wide used knowledge base systems in daily applications. By its ease of use systems can be developed rapidly [54]. Therefore it is a good candidate for the reasoning system of the AIBO watchdog. Time can be saved for the other components in the reasoning system.

Case based reasoning

Case based reasoning uses its experiences that it has gained during previous interactions to decide the output based on the current input. The most likely case will be selected and its output will be determined. Case based reasoning is therefore also a good candidate for the reasoning system of the AIBO watchdog.

Bayesian network

Bayesian network deals with the uncertainty of the inputs. Due to noise in sensors and ambiguity in the real environment uncertainty of the inputs will certainly arise. In this aspect Bayesian network is one of the best reasoning system for uncertain environment, but due to its complexity it is hard to realize.

Neural networks

Neural networks will be trained with the training samples. These samples have to be developed to train the neural network. After successful training it will usually work correctly when the inputs do not differ from the training samples. The user is not able to follow the reasoning of the network and there is no guarantee that the network will always produce the correct output in all situations. Modifying the reasoning can not easily be achieved.

For showing the correct working of the developed architecture it is necessary to understand the reasoning process. Expert systems, case based reasoning and Bayesian network provides the transparency of the reasoning process. For creating the Bayesian network many researches have to be conducted to obtain the probabilities of the occurring events. Since this thesis project does not only consist of a reasoning system, but also the other components in the reasoning architecture, it is wiser to choose another reasoning system which fits in the time constraint. The predecessor of the AIBO watchdog project has chosen for the expert system. The author will continue with this expert system and some case base reasoning concepts will be added in the expert system. The reasoning system is the so called dynamic scripting approach. In the next future the Bayesian network is on the planning.

2.7. Memory of Reasoning Systems

For creating an intelligent reasoning system it is required to be able to reason about the current situation based on nothing or based on the past. The last option can provide the AIBO watchdog more intelligence and behaving more naturally. The information of the past can give human beings a global illustration what they can expect at the moment. It allows the AIBO to reflect on the past and anticipate the future [20]. How can memory help us to remember the current events? To answer this question we need to know the basic knowledge of remembering.

Varieties of remembering

‘Memory’ is a label for a diverse set of cognitive capacities by which humans and perhaps other animals retain information and reconstruct past experiences, usually for present purposes. Human’s particular abilities to conjure up long-gone episodes of their lives are both familiar and puzzling. People remember experiences and events which are not happening now, so memory seems to differ from perception. They remember events which really happened, so memory is unlike pure imagination. Memory seems to be a source of knowledge, or perhaps just is retained knowledge. Remembering is often suffused with emotion. It is an essential part of much reasoning [19].

- **Procedural memory**

  Philosophers' ‘habit memory’ is psychologists' ‘procedural memory’, a label for embodied skills such as typing, playing golf, using a knife and fork, dancing, or solving jigsaw puzzles. People naturally refer to procedural memories with the grammatical construction ‘remembering how’ [19].

- **Semantic memory**

  ‘Propositional memory’ is ‘*semantic memory*’ or memory for facts, the vast network of conceptual information underlying human’s general knowledge of the world: this is naturally expressed as ‘remembering *that*’, for example, that Sony created AIBO in 1999 [19].

- **Episodic memory**

  ‘Recollective memory’ is ‘episodic memory’, also sometimes called ‘personal memory’ or ‘direct memory’ by philosophers: this is memory for experienced events and episodes, such as a conversation this morning or the death of a friend eight years ago. Episodic memories are naturally expressed with a direct object: I remember our argument about Descartes yesterday, and I remember my emotions and my bodily sensations as we talked. Such personal memories can be generic or specific, and they can be memories of more or less extended temporal periods [19].

Both semantic and episodic memories, whether linguistically expressed or not, usually aim at truth, and are together sometimes called ‘declarative memory’, in contrast to nondeclarative forms of memory, which don't seem to represent the world or the past in the same sense [19].

2.8. AIBO World Environment

In this section the global world model of the AIBO watchdog and the way to interpret the world environment will be presented.

2.8.1. World Model

The world model of AIBO watchdog describes the interaction processes between AIBO and its environment. AIBO sensors and actuators provide the capability to interact with its environment. Figure 2.16 illustrates a global real world environment of the AIBO watchdog.

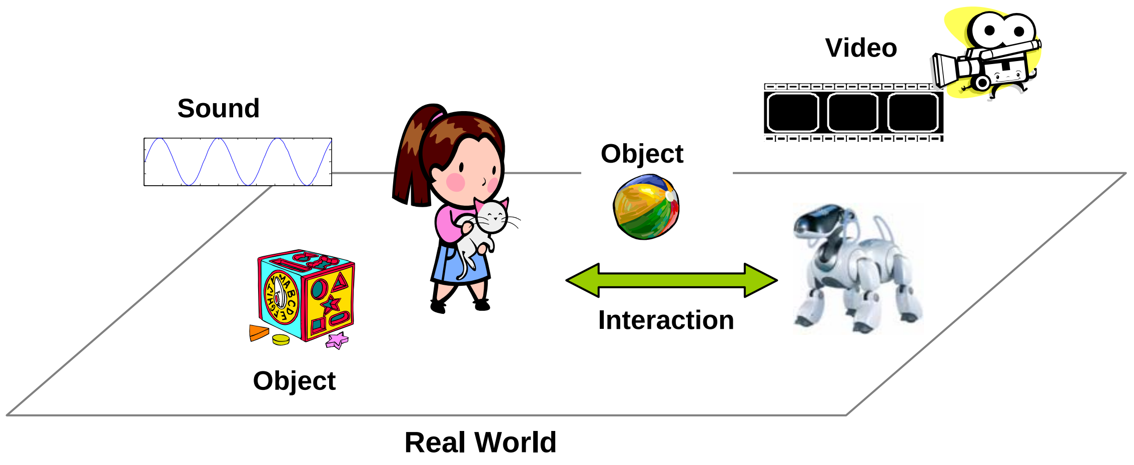


Figure 2.16: A global representation of the real world environment.

AIBO watchdog is able to interact with its owner and the objects in its environment. The features from the environment e.g. sound and video can be perceived by the sensors of AIBO watchdog. Based on these features AIBO watchdog will react to them. Reactions are executed by its actuators. Table 2.1 shows the world model of the AIBO watchdog.

Table 2.1: World model AIBO watchdog.

	List of features	Functionality	Relation
AIBO watchdog	4 legs	able to move/walk	object collision avoidance
	mouth (speaker)	able to bark	people notification by sound
	eyes (camera)	able to see objects	object recognition
	ears (microphone)	able to hear sounds	sound recognition
	wireless connection	able to send digital messages	people notification by digital messages
	Face (many colored leds)	able to show emotion	people notification by emotion

2.8.2. Interpretation of World Environment

The ways that robots use to understand its world environment is comparable with the human beings. Objects recognition in its environment can be achieved with image sensors, cameras, and sound sensors, microphones. By comparing the features of the objects in its environment with the features in its database robots are able to identify the objects [23]. The global idea of this approach is illustrated by Figure 2.17.

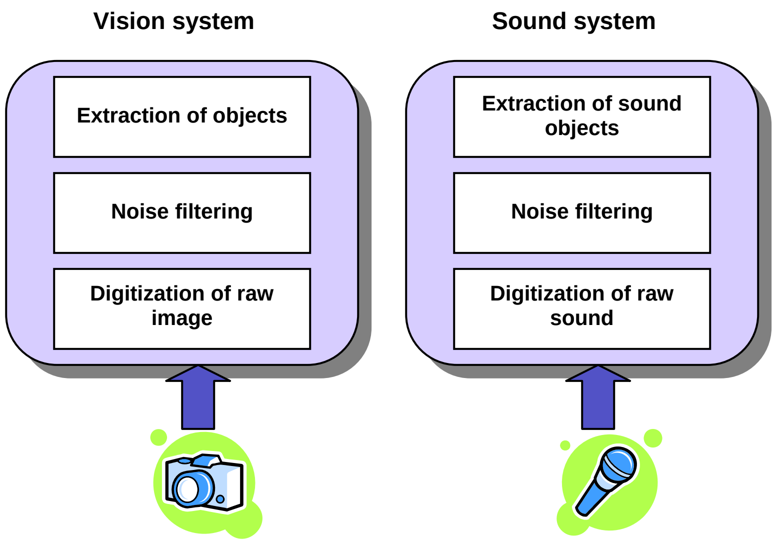

Vision system
Sound system
Extraction of objects
Noise filtering
Digitization of raw image
Extraction of sound objects
Noise filtering
Digitization of raw sound

Figure 2.17: Working of object recognition with sound and vision.

The received analogue signals from the sensors will be digitized first before other software techniques can be applied. The result of the digitization will be filtered by a filtering technique. This filtering technique can be built in the hardware or using a software to remove the noise. Using the clean result of the raw image objects on the image can be extracted by an object extraction program

[22]. The working of the sound object recognition system is the same as the vision system. Instead of raw image data the input is using the raw sound data.

2.8.3. Concept for World Environment Interpretation

The environment is an open system which implicates that it can have infinite objects. Therefore it is impossible to recognize all possible objects in the world environment. Every object has its own unique features: color, shape, material, size, weight, smell, sound etc. This is the only way for human beings to distinguish a certain object from the other ones. This theory also applies for the robots. By perceiving the features of the objects using the sensors the robots are also able to classify the objects. Figure 2.18 illustrates the interpretation concept of robots with its environment.

The perceived features will be interpreted by the robots and objects can be recognized. The interpretation process can be a matching of perceived features with a features database. Because of the ambiguity of objects in the world, it is not always possible to recognize the object correctly. For example an orange and a ball both have the same color, shape and size, but the other features are different. The issue is whether they are detectable by the sensors or not.

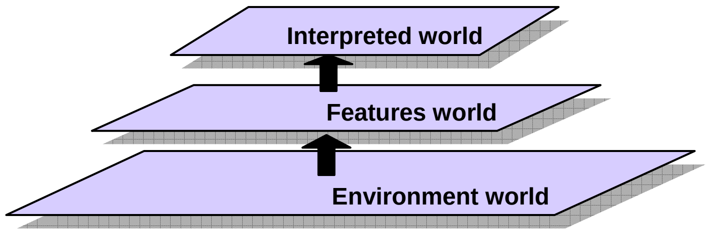


Figure 2.18: Interpretation process of the world environment.

3 Related Research

"The Past: Our cradle, not our prison; there is danger as well as appeal in its glamour. The past is for inspiration, not imitation, for continuation, not repetition."
Israel Zangwill

In this chapter the projects that are related to the AIBO watchdog will be presented. First all related projects concerning the AIBO robot dog will be discussed and thereafter useful papers from the literature research that can improve the AIBO watchdog will be presented.

3.1. Related Work

Since the launch of the first AIBO robot dog 1999 by Sony a lot of scientists are using this robot dog to conduct research experiments. Because its affordable price and good possibilities it is the most popular robot used in research.

3.1.1. AIBO Soccer Robot Competition

Robot soccer competition, RoboCup is an annual event that will taken place somewhere in the world [47][48]. This year, 2006, this event has taken place in Germany. Many teams from everywhere around the world has gathered in this place and compete with each other for the honorable first place. The soccer teams consist of 4 AIBO robots which will play in a 6 x 4 m field. In a strategic point of view coordination and cooperation of AIBO is the key factor to win the competition. On the other hand in the technical point of view ball control and shoot techniques are also an important issue.

The reasoning system of the soccer robots must deal with object recognition and motion expectation. For strategic purposes it also has to deal with positioning of its own AIBO for smoothing their offense or defending the attack of the opponents.

3.1.2. Fusing Speech and Face Recognition on the AIBO ERS-7

This project is about the fusing visual and audio recognition on the AIBO robot dog [64]. The AIBO is communicating wireless through a TCP/IP connection to a remote computer which manages face and speech recognition of a person in its environment. Both visual and audio data are sent by the AIBO to the computer which analyses them and gives a feedback to the robot how to behave according to the result of the calculation.

The author has used two modalities to recognize persons: vision and sound. By fusing these 2 modalities the author tried to achieve better recognition result. A face recognition algorithm designed by the author was implemented and tested. For the sound part a sound localization module was used which has been developed by the MMI Department of TUDelft and has proved its value at the Robocup in 2004.

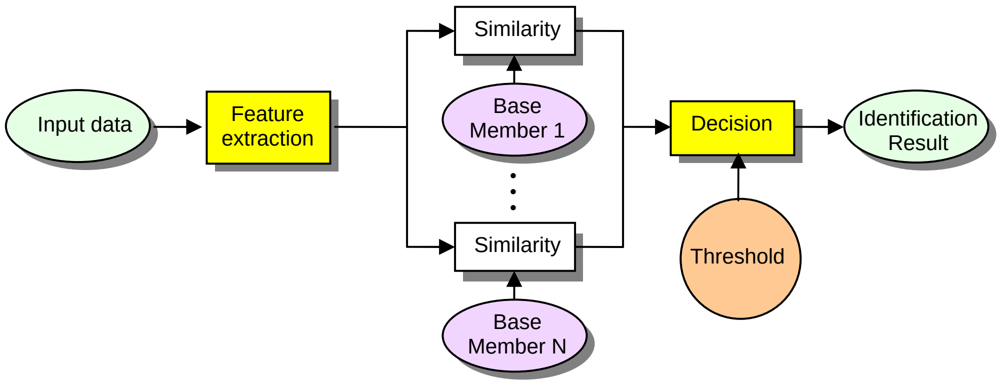


Figure 3.1: Recognition model.

Figure 3.1 shows the recognition model of the designed AIBO. First the features of the input data will be extracted and thereafter be matched with the features in the Database. Depending on the matching differences a decision will be made and the AIBO will change its behavior.

Because of the time constraint and the use of Matlab files, this approach was realized with Tekkotsu and URBI, using the client/server architecture. Therefore a normal pc is needed to process the calculations. As many papers have also indicated about image recognition it is very hard to deal with the light conditions in the environment. When the light conditions are very weak or very strong, the color table is completely different from normal circumstances. Therefore the face recognition algorithm works only acceptable.

The used approach was based on static images from the camera. Streaming video may provide better results. The image identification algorithm seems to work not fast enough to keep the same pace as the image input. The sound identification part resulted to an acceptable success rate, but it is very dependent on the quality and duration of the referenced sounds. It is required to have referenced sounds which have more voice characteristics. During its test phase silences give the wrong identification. This multimodal approach for the face recognition can be a good additional functionality for the AIBO watchdog. Being able to recognize the owner makes the watchdog more social and it also gives a good feedback to the owner.

3.1.3. Autonomous AIBO Watchman

In this project the author tried to let the LEGO robot collaborate with the AIBO robot and a PC in between, which functions as a communication server [62]. A whole risk and cost analysis is made to compare the lost when using the AIBO as a watchman (guard) instead of human beings. Not really surprisingly the AIBO was chosen as the most suitable candidate to do the job. Especially the low operational cost and less threatening when heavy weapons were used, was the big advantage of the AIBO.

There are a lot of diagrams which are modeled for the AIBO watchman. A big part of the programming source code is also included in the report. The author used Tekkotsu to program the AIBO watchman, because there was no other framework available at that moment.

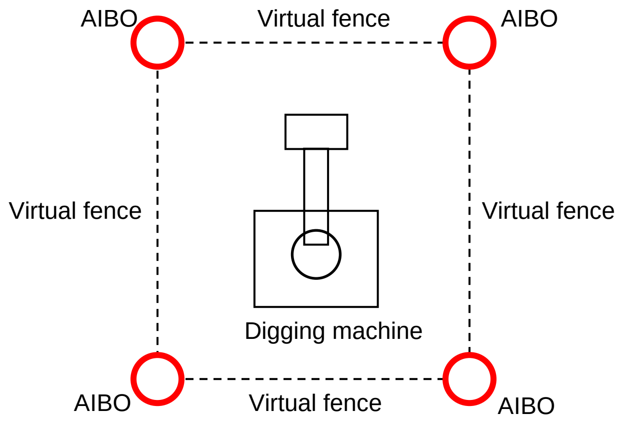


Figure 3.2: Model of the digging machine and watchmen.

The mission of the project was to use the AIBO robot dogs as guards to guide and support the Lego digging machine. Figure 3.2 shows the model representation of this situation. Everything that crosses the virtual fence will be noticed by the AIBO dogs. As a result the digging machine can operate in a safe environment. The functionality of the AIBO 'watchman' was modeled very limited: walk forward, detect a wall, detect a pink ball as an intruder and respond to it. These are the most basic operations that an AIBO watchman can carry out, but not all functionalities has been implemented and tested.

3.1.4. AIBO Watchdog and AIBO Companion Dog Project

At the University of Technology Delft in the MMI department there were also 2 projects conducted concerning the reasoning system of the AIBO robot dog. The first project is concerning the personality model of a companion robot dog. Its main purpose was developing a reasoning system which can interact with humans by showing emotions and actions [49]. The second project is concerning the AIBO watchdog [40]. These 2 projects will be described in this section.

AIBO Companion Dog

In this project the author describes a complex set of models and architectures that allow the embodiment of an emotionally intelligent robot in interacting with other humans or robots. The robot used for deploying these ideas and concepts is the AIBO robot dog.

The author has designed a new emotion reasoning model for the AIBO companion robot which can give the AIBO companion robot its own personality. Figure 3.3 shows this reasoning model.

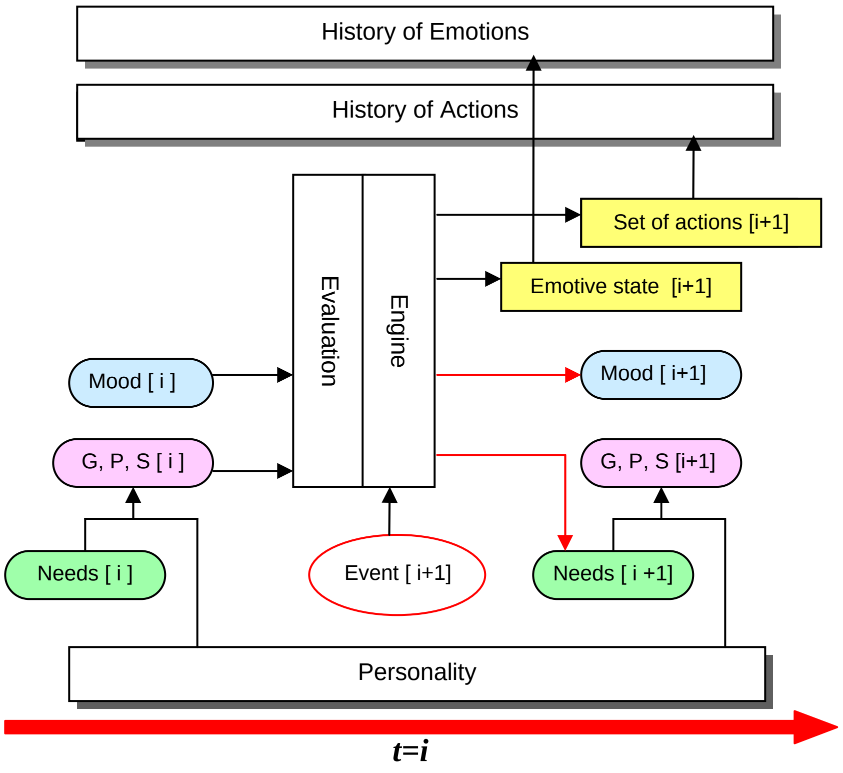


Figure 3.3: Emotion reasoning model.

Depending on the current mood, needs and other environment variables, new mood, needs and actions will be inferred. After a platform comparison the author chooses URBI as programming platform and Jess as rule based Inference Engine. All programming models and details were given. Most of them were implemented in the AIBO at the end. This report can serve as the base ground for the AIBO watchdog.

Due to time constraints the only working components were the emotion reasoning functionality what the main purpose was. As the author also has concluded and what the AIBO watchdog need is a memory and history module. Based on the history a more sophisticated action can be inferred from the rule based Inference Engine.

A multimodal approach was desirable. By integrating vision and sound more accurate results can be achieved, but problems like synchronization will arise. The emotion reasoning model can be a part of the AIBO watchdog, when the basic functionalities of a watchdog are operational.

AIBO watchdog

The second project was developing an AIBO watchdog. AIBO thought as a companion dog that entertains people and makes people smile could be also programmed to do more useful things such as protect us and announce us when it sees an intruder in the home environment. AIBO could be a dog, a companion and a surveillance camera and it also has the capacity of being on duty day and night, 24 hours of 24 hours, 7 days a week.

Recent researches have been done on transforming the AIBO into a watchdog. These researches transformed AIBO into a simple camera that has the capacity of barking at the moving objects and saving an image of the moving object which could be later seen by the owner.

This project inspires to “train” AIBO ERS-7 dog to be a watchdog. The AIBO will be able to detect motions and sounds, to bark and save images of moving objects, engage into investigations, and moreover will be able to “see” the intruder. It could even be used as a smoke detector: if AIBO "smells” smoke it starts exploring to see if the house is on fire. AIBO watchdog will send an alarm via his wireless communication network if there is any threat. The AIBO watchdog will act as a live trained dog and maybe in time it will replace the real ones.

But due to time constraint the system lacks of a good reasoning system. A very simple reasoning system was used and detecting relevant objects is not possible. The author has chosen for a static environment which does not provide good interactivity for the AIBO and its environment [40].

Therefore the author has chosen to complete the AIBO watchdog project that his predecessor has left behind. A complete new design will be developed and existing models will be improved to be a part of the new design.

3.2. Literature Research

Before the design phase a lot of researches have to be conducted that are related to the AIBO watchdog. The existing theories and algorithms can be refined and used for the AIBO watchdog. The relevant research can be an inspiration source for the design of the reasoning system. The working of the AIBO watchdog can be divided into 3 main aspects: use of multimodalities, path planning and world modeling, internal components. Each of these aspects will be discussed in detail in this section.

3.2.1. Multimodalities

In the case of AIBO watchdog it uses the image sensor, sound sensor and distance sensor. Benefits and drawbacks of the use of multimodalities have to be researched and methods to improve the concurrent working of these sensors. This section will discuss 2 papers that have a high potential to improve the AIBO watchdog.

Purposeful Perception by Attention-Steered Robots

In this paper the authors have proposed a new heuristic in image perception [56]. Sometimes it is not necessary to be able to perceive everything in your view and remind them; by untying your focus you can get more relevant information.

In purposeful perception you select only the most important object that you want to perceive in your view. When you do not see that object in the current view, it is not necessary to memorize the other objects in the current view. They are not the primary target, but secondary target. Secondary targets are only useful when the primary target exists. Figure 3.4 shows the architecture of attention steered vision.

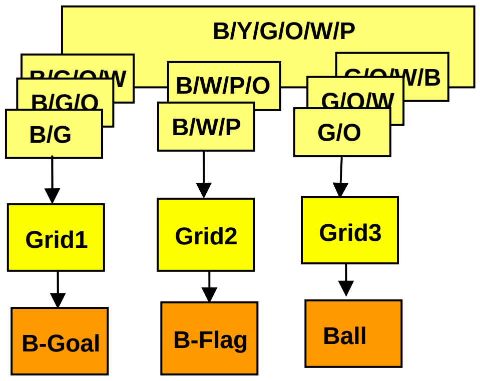


Figure 3.4: Architecture of attention steered vision.

When using this heuristic you can use the shape or the color of the primary object. This heuristic is very useful when you are looking for something, e.g. looking for a pink ball. It does not matter what kind of color you see around you, if it is not the pink color, you are not interested in it. That is also the reason that you do not need to remember the other color that you saw. As the architecture also shows, if the AIBO does not see the color (B)lue, it means the AIBO is not facing the B-goal or B-flag.

For the AIBO watchdog this is a useful heuristic which can speed up the processing time and simplify the programming code. Since the watchdog is only looking for suspicious and abnormal situations.

Audio-Visual Flow – A Variational Approach to Multi-Modal Flow Estimation

In this paper the authors have designed their own framework to estimate the audio-visual flow field of a moving audio source [57]. By using the multimodal approach, in this paper vision and sound, it is much easier to track a moving audio source object with the camera. The authors explained with their framework and formula how to calculate an audio flow field. Using this audio flow field we can match it with the images of the camera. When something suddenly goes in front of the camera and blocks the moving audio source object, this approach is still able to follow the moving audio source object with its camera using their sound localization theory. Therefore the camera will not lose track of the audio source object, when the blocking object are moved away. Figure 3.5 and Figure 3.6 illustrate this concept.

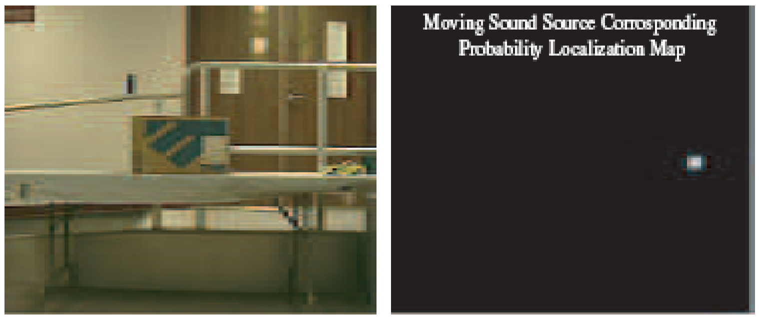

Moving Sound Source Corresponding
Probability Localization Map

Figure 3.5: Left image: image from the camera, moving yellow car.
Right image: sound source localization map.

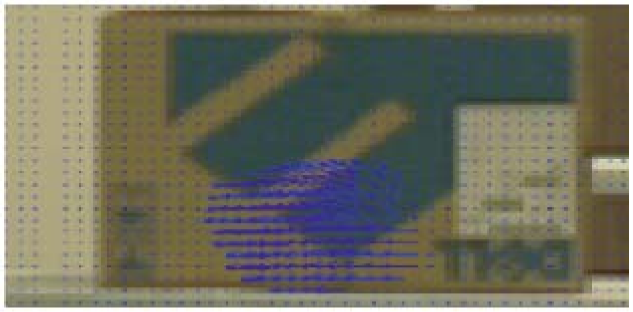


Figure 3.6: Sample frame with flow field of the fully occluded moving sound source.

This paper offers only a solution for one moving audio source object. Multiple moving audio source objects will be a big problem, especially when they have the same audio output. Another problem which this paper has ignored is the background sound. In many cases the background sound can be minimized with a sound filter, but they have not tested their framework under these circumstances. Therefore this framework and calculation will perform its best in a very quiet environment with only one moving audio source object.

3.2.2. Path Planning and World Modeling

AIBO watchdog has to be able to move freely in its environment; therefore good path planning algorithms are necessary. Prior knowledge of the environment can simplify the path planning system, but sudden changes in the environment must also be taken into account. In this subchapter 3 papers regarding this topic will be discussed.

AIBO Programming Report of Group Vision1b

This paper discussed several possible models for representing the world environment: grid base, graph base, Kalman Filters, particle maps and a mix of these models [65]. All approaches have its own advantages, so it depends on the kind of information that you have from the environment to choose which approach is the best to use. After having the world model, the robot has to navigate to its destination. When we are having a holonomic robot, a robot that has to turn something, e.g. the steer axes, to determine the path direction; it is needed to take the movements restrictions of the robot into account. The planner component of the robot has to deal with the movements restrictions and define the correct actions. For the path planning component the paper discussed 2 planning algorithm: visibility planner and probabilistic planner. Both planners used graphs to find the paths. The last one is not optimal, but it is simpler to implement.

For static environments there are already some path planning algorithms available, but in dynamic environments it is hard to plan your path. Many objects are moving constantly. The author made this research for the soccer AIBO; therefore the planner will not be that difficult. There are not much obstacles on the soccer field. Since the AIBO watchdog has to find its way in a normal house, there will be lots of obstacles on its way. This paper functions as an information source for the path planning algorithm and world modeling of the AIBO watchdog.

Improving AIBO with Artificial Intelligence Technique

This paper proposed an interesting functionality for the AIBO dog and a solution to achieve that [58]. Many dog owners have already done it before with their real dogs. The words ‘AIBO, come here’ should be familiar with most people. The offered method for this problem is very logical. At first the AIBO has to recognize the words, ‘AIBO, come here’, and locate the voice direction. Thereafter there are 3 new basic functions needed: map construction, room recognition and way finding. The automated way to construct the map requires an object recognition algorithm with simple shape and color recognition ability.

The proposed idea is the semi-automatic approach, just like showing your house to your guests. The owner can take the AIBO to each place of the house and tell where it is. This way the AIBO can draw a graph of the whole house. Figure 3.7 illustrate this concept.

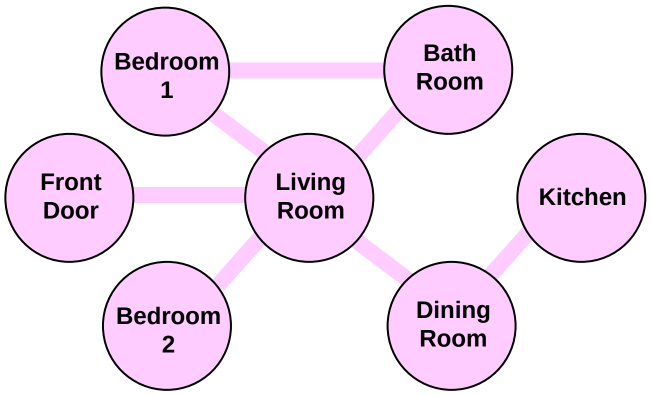


Figure 3.7: Map of a house represented with a graph.

To recognize each room it is recommended to put the tags on the doors which are easier to locate. Instead of ID tags the author recommended to use Cyber Codes for tagging the rooms. This 2D-tagging system offers more information than ID Tags, e.g. position and orientation. After having a graph of the whole house it is not hard to find the way from one room to another one. A simple depth first search algorithm can simply do the job.

The proposed functionality of the AIBO, ‘AIBO, come here’, is a part of the AIBO watchdog. The proposed solution to realize this functionality is very smart. The whole idea is very subtle, but the details have to be worked out to realize this idea.

Obstacle Avoidance and Path Planning for Humanoid Robots using Stereo Vision

The authors have applied stereo vision to estimate the floor distance and obstacles on the floor [59]. As a result a biped humanoid robot can walk safely on the ground and avoid obstacles on its way. First the two images of 2 vision systems have to be combined to a bigger image by using block matching. Obstacles with lots of textures will be recognized as obstacles. While scanning the area with both cameras on its head, a grid map will be made. The whole area will be divided into grid lines. Not scanned area will be represented with a particular color on the map. Obstacles and floor will be given with another color. During walking around its environment a real time grid will be made that is used for path planning. A simple A* search algorithm can find a path to move to its destination. This concept is illustrated in Figure 3.8.

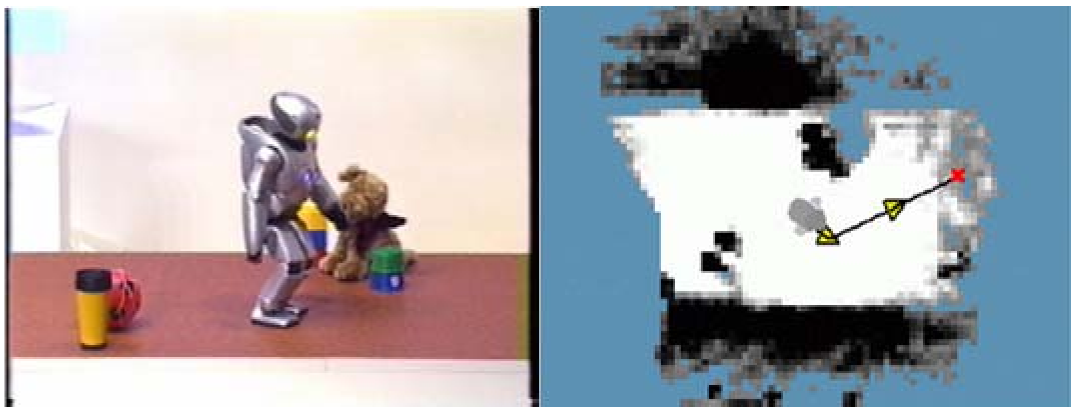


Figure 3.8: Right image shows the occupancy grid and planned path of the left original image.

As the authors also have mentioned this approach can only work when the environment contains enough textures. Using the approach the biped robot is able to move around in a static environment. The authors have not given a solution to the problem when the environment is not static, but dynamic. So when a moving object is running around, shall we take another path or walk carefully without standing on it, but this is the first step to move to that goal. A global architecture was given to use this stereo vision to avoid obstacles and for path planning. This global architecture is illustrated in Figure 3.9.

Kinematics component reads all the joints sensors and image sensor. The Plane Extractor component extracts the objects of the image and let the Occupancy component update the occupancy grid. As a result the path planner can find an obstacle free path to its destination. This paper can serve as a baseline for obstacles avoidance and path planning of the AIBO watchdog, however the physical possibilities are not the same; AIBO has only one camera.

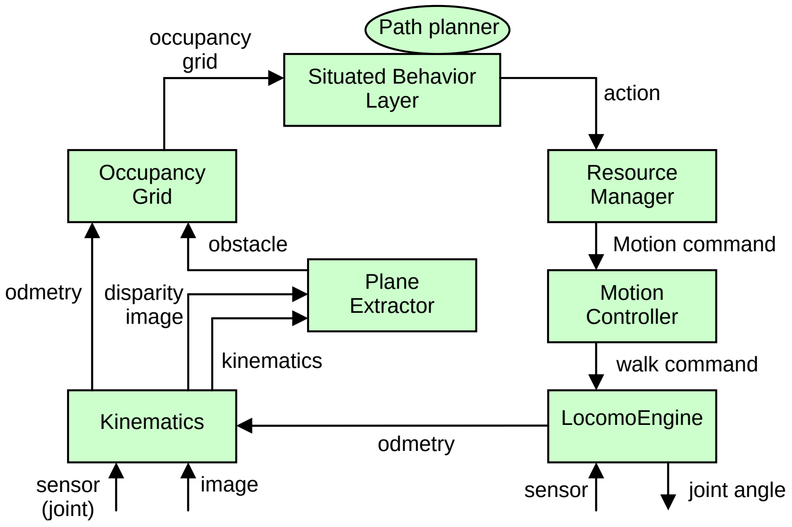


Figure 3.9: Global software architecture of obstacles avoidance and path planning.

3.2.3. Internal Components

Internal components are concepts components that are necessary to improve the intelligence of robots. By providing the AIBO watchdog a short term memory component it is able to recognize events in time space. A long term memory component can provide storage of its learning during its exploration. Task planner component will choose the right task to be executed at the right time. In this section 3 papers will be discussed.

A Robot that Reinforcement-Learns to Identify and Memorize Important Previous Observations

The author has designed an approach to extract the important events from the previous observations and use them to make a decision later on [60]. Situations where the environment is fully observable one can decide its output immediately, but in partially observable situations there is a short term memory needed to remember the important events to search for the appropriate output. The author has set up a test case for this theory.

First the events will be extracted with an unsupervised learning method, ARAVQ (Adaptive Resource Allocation Vector Quantization). Thereafter the events will be put in the Reinforcement Learning (RL) system. The Long short term memory (LSTM) is an important part of the RL system. Events that are useless will be discarded by this component. Events that are no more needed will be removed from the LSTM to prevent overload. Only the useful input and output will be memorized. Figure 3.10 shows the architecture of the RL-LSTM network.

The data will be gathered with online exploration. After many iterations the gathered information will be used for the offline learning. As a result the author achieved very good result with this model. Since the AIBO watchdog does not have a full view of the whole environment, this approach forms the base idea to let the AIBO see the relationship between two or more related events.

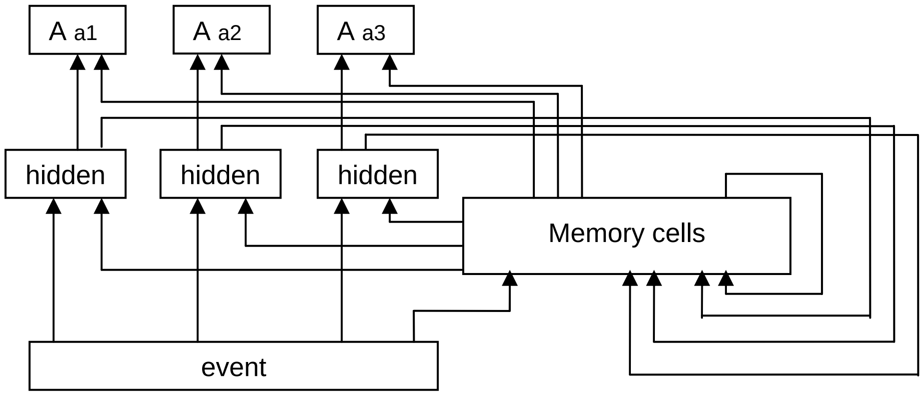


Figure 3.10: Architecture of the RL-LSTM network.

Autonomous Rovers for Mars Exploration

This paper explained what a Mars rover robot has to be [61]. It has to be a robust semi-autonomous robot which can deal with all kind of environment. Self-repairing and charging are basic operations that it can operate. Navigating around the Mars-planet and decision making when in danger are also a part of the basic operations. The architecture that they have developed to meet the requirements is also promising. Figure 3.11 illustrates this architecture.

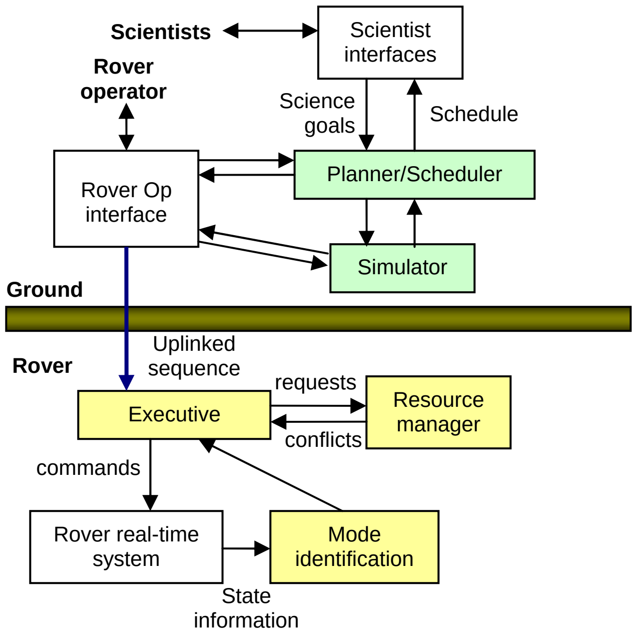


Figure 3.11: Mars rover architecture.

This architecture has a planner which deal with the priorities of its missions. The Rover operators from Earth can set the priorities when needed. The resource manager inside the rover robot, the mode identification object and the Executive object are the intelligence of the Rover robot. A more detailed architecture of the inside of the Rover robot was not provided, but the important components were explained with figures. However this architecture is not fully autonomous, it can serve as a good inspiration source for the design of the AIBO watchdog.

Find Kick Play - An Innate Behavior for the Aibo Robot

The author proposed to use innate behavior to let the AIBO develop and learn about its environment [63]. The innate behavior that was designed and implemented is finding the pink ball and kicking it to the goal. A lot of basic functions were needed and most of them can be used for the AIBO watchdog. The common problems were also discussed and solved partially.

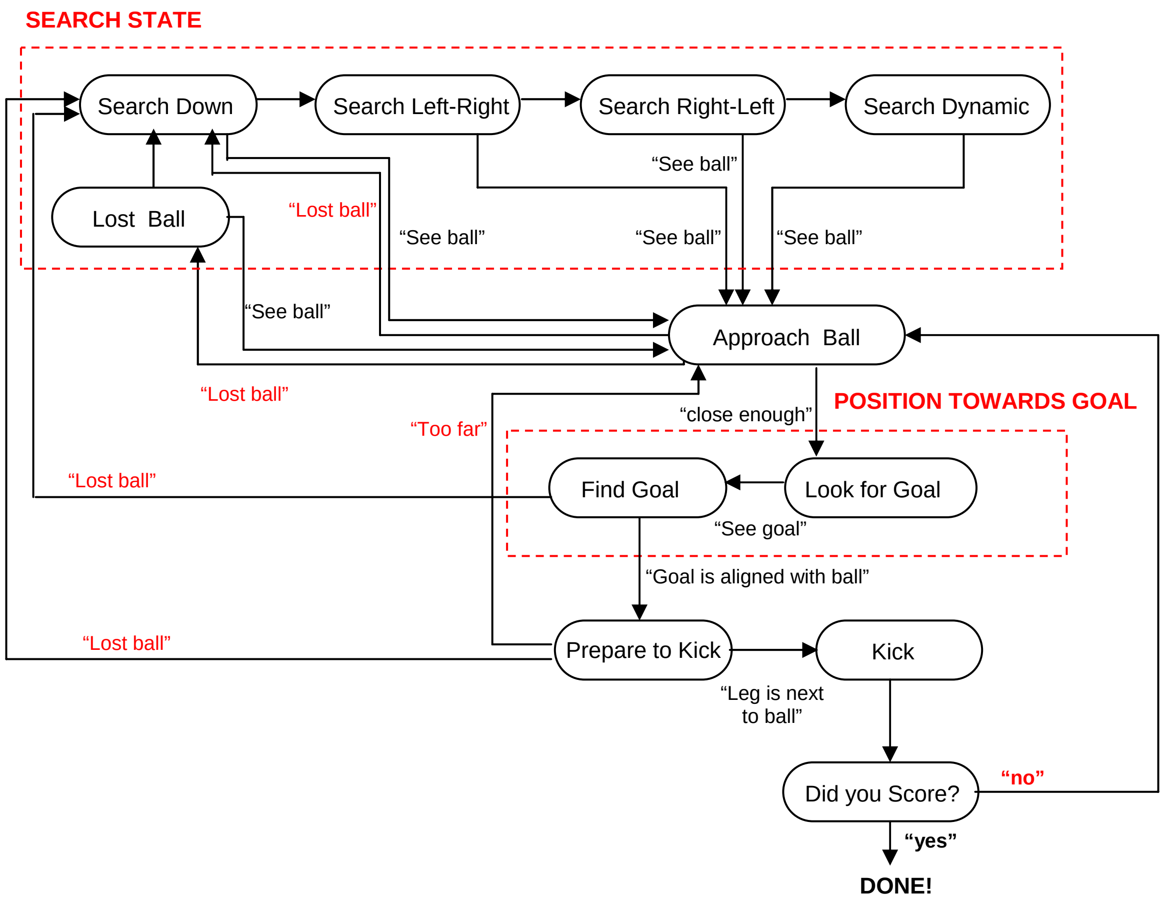


Figure 3.12: States of the Finite State Machine of the search and kick behavior.

Problems like object recognition, setting up a connection with the server, stable walking functionality, stable camera input, searching for the ball object, etc. were discussed and resulted to some useful advises. The author has used a Finite State Machine (FSM) to model the innate behavior. Since Finite State Machine is designed modular, it is very easy to add or create new behaviors with this model. Figure 3.12 illustrates the states of the Finite State Machine of the search and kick behavior.

All of these functionalities were realized with the Pyro AIBO Controller connected with the Tekkotsu server. The author has succeeded with converting his model to working implementation. Detailed programming code was not given why this paper resulted to only some useful hints and working recommendations. The result of this literature research is summarized in Table 3.1.

Table 3.1: Summary of the result of the literature survey about the concepts which can be used for the AIBO watchdog.

Use of multimodalities
• Use purposeful perception to look for abnormal objects and situations. • Use audio-visual flow estimation to follow suddenly hidden moving sound objects.
Path planning and world modeling
• Use graph representation for the home environment. • Use occupancy grid for individual rooms. • Use Cybercode tagging on the doors for room identification.
Internal components
• Short term memory components. • Task planner and resource manager. • Innate behavior component.

4
AIBO

"Anything you can do needs to be done, so pick up the tool of your choice and get started."
Ben Linder

In this chapter the most important issues regarding the AIBO robot dog will be discussed. First the AIBO hardware will be discussed and thereafter the software environment. The working of the AIBO concept will follow and at last the functionalities of the AIBO watchdog regarding the hardware specifications will be presented.

AIBO is a robot dog which is developed by Sony as an entertainment robot [7]. There are lots of variants of the AIBO robot dog. Every new one gives more possibilities than the previous one. The AIBO that this project uses is the revised version ERS-7 which is launched in the winter of 2005. The specifications of AIBO-ERS7 are shown in the table below.

Table 4.1: Specifications of the AIBO-ERS7.

CPU	64-bit RISC Processor
CPU clock speed	576 MHz
RAM	64 MB
Program media	Dedicated AIBO robot "Memory Stick™" media
Moveable parts	Head – 3 degrees of freedom Mouth – 1 degree of freedom Legs – 3 degrees of freedom x 4 Ears – 1 degree of freedom x 2 Tail – 2 degrees of freedom (Total 20 degrees of freedom)
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
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	1.6 kg (including battery pack and “Memory Stick™” media)
Operating temperature	5°C to 35°C (41°F to 95°F)
Operating humidity	10% to 80% (no condensation)
Operating wet-bulb temperature	Max. 29°C (84°F)
Storage temperature	-10°C to 60°C (14°F to 140°F)
Storage humidity	10% to 90% (non condensation)
Storage wet-bulb temperature	Max. 29°C (84°F)
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)

4.1. AIBO Hardware

Figure 4.1 and Figure 4.2 shows the AIBO features respectively from the front view and the rear view. AIBO receives the information from its environment through its 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 [7].

The equipments of the AIBO to show its reactions are: speaker on its chest, LED Lights (on face, ears and back) and a series of movable parts: head (3 DOF1), 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 connection 802.11b and a blue LED to show its status [7].

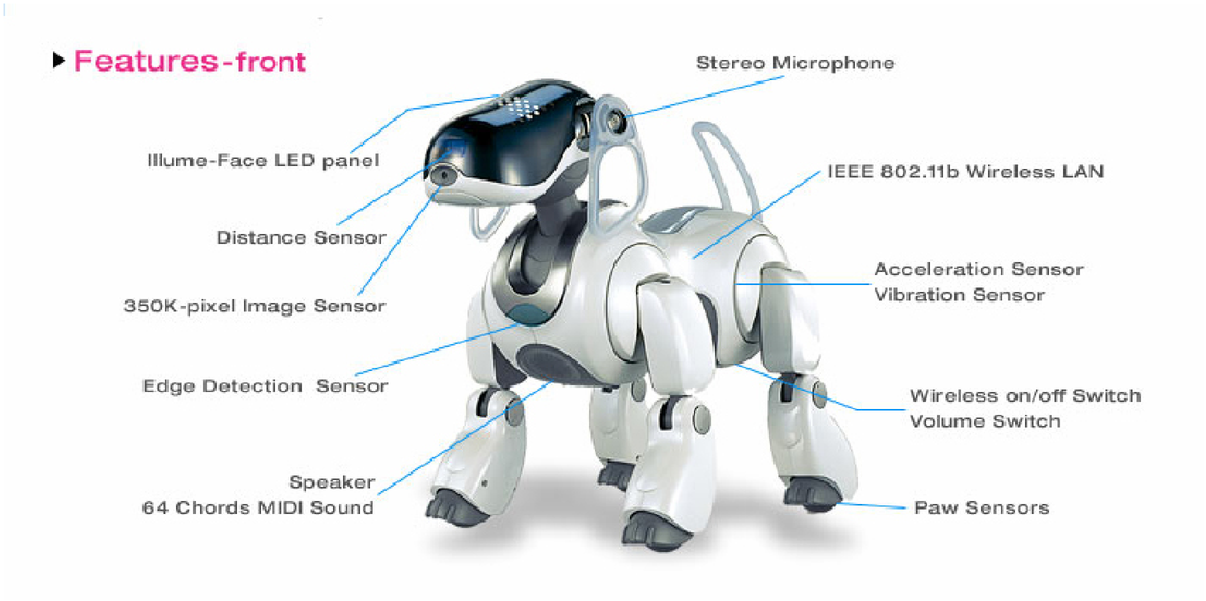


Figure 4.1. AIBO sensors and actuators – front view.

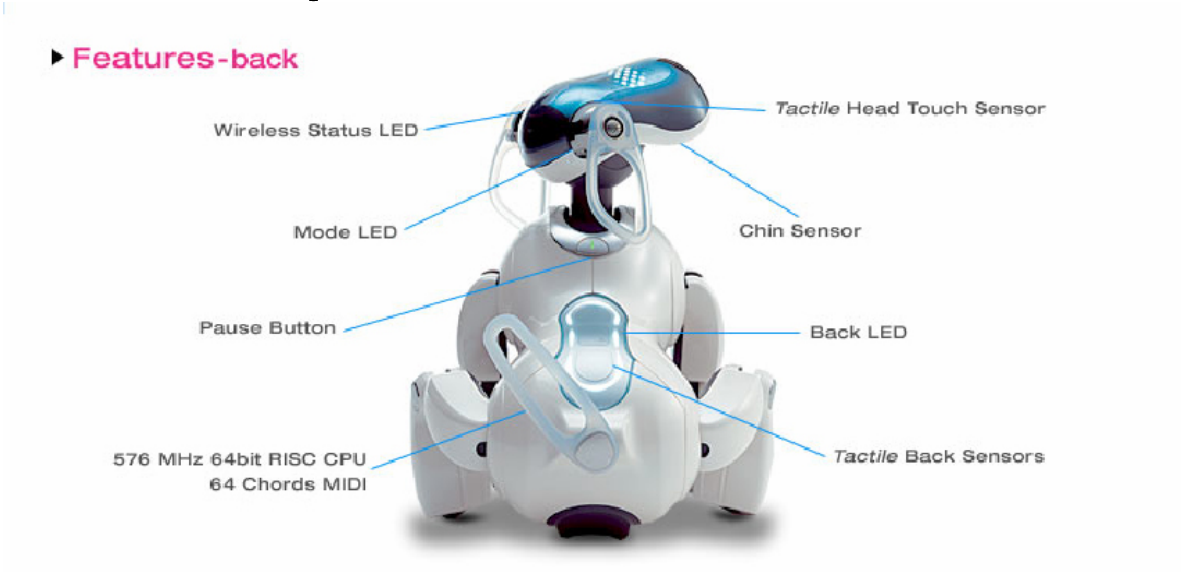


Figure 4.2 AIBO sensors and actuators – rear view.

4.2. AIBO Software

OPEN-R is promoted by Sony as the standard interface for its entertainment robots systems, such as the AIBO. Figure 4.3 shows the hardware/software architecture of the AIBO.

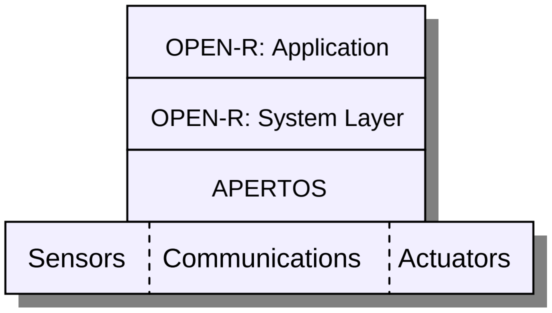


Figure 4.3: Hardware/software architecture of AIBO.

OPEN-R system layer is built on top of the operating system APERTOS. OPEN-R system layer offers the access to the software, hardware and network capabilities of the AIBO. In the OPEN-R application layer Sony has developed advanced features for the AIBO, such as face recognition, voice recognition, obstacles avoidance, etc. Since Sony does not provide the documentation about the application layer, there is only documentation about the system layer. Therefore designing and implementing a new model can only be realized with help of third parties software which interact directly with the OPEN-R system layer.

4.2.1. Apertos

APERTOS is an object-oriented embedded operating system based on meta-level architecture. Many of the APERTOS' design concepts have a heavy weight in the way AIBO is programmed with OPEN-R. Everything in APERTOS is an object. Each object encapsulates the state, methods which access such state, and a virtual processor which executes its methods.

The communication inter-objects is made by message passing and the object execution is guided by events. After the initialization, an object uses to be idle. When an object wants to communicate with others, it sends a message, writing the data in shared memory and sending an event to the destination object, which will eventually be activated, it will read the data and handle it depending on the message type has arrived. These events have some assigned priority so that it can be distinguished between an ordinary event from a hardware interruption and other events.

When an object is performing an operation (a method typically) it will not be interrupted by an event or interruption until the operation is finished to avoid race conditions. It will achieve this by interruption masking. The message that can not be manager immediately is stored in a message buffer allocated in shared memory. Although all the objects share the same memory space, no object can overwrite any data belonging to other object. The only exception is in message buffer case. The message delivered to other object is allocated in a shared region in memory. APERTOS does not provide a transparent way to protect this memory. It only provides a counter for references to a memory region. The objects use a meta-hierarchy of meta-objects to define its behavior. The set of meta-object that a object uses is called meta-space. If an object, for example, wants TCP/IP communication and its meta-space do not support this, the object can migrate to other meta-space that support this kind of communication [50].

4.2.2. Open-R system Layer

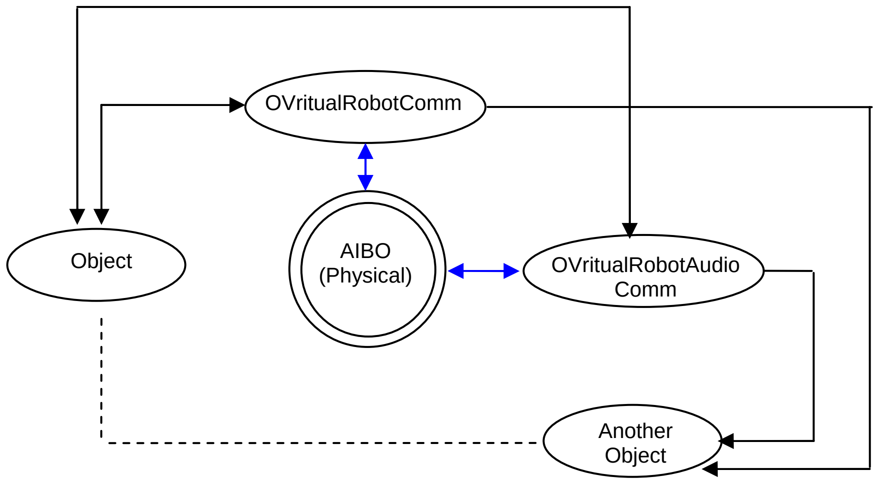


Figure 4.4: Open-R System layer overview.

Figure 4.4 shows an overview of the OPEN-R system on AIBO. The system contains several objects that communicate with each other by means of message passing. The term “object” is not the same as the term “Object” as we know it from the Object Oriented programming paradigm. In Open-R, an object must be interpreted as an executable program. In this sense, OPEN-R is a system in which several objects run concurrently and communicate with each other through message passing. The object sending a message is called a subject and the object receiving the message is called the observer. A single object can be subject and/or observer depending on the situation.

The OPEN-R system contains two special objects, OVirtualRobotComm and OVirtualRobotAudioComm, these are the only two objects that can directly access the physical AIBO (sensors, joints etc). From a programmer’s point of view, both objects can be regarded as any other normal object [51].

Object OVirtualRobotComm deals with the joints, LEDs, sensor values and image capturing. On the other hand Object OVirtualRobotAudioComm deals with the microphone and speakers of the AIBO. These objects will be steered by the commands of the application layer.

4.2.3. Open-R Application Layer’s Software Development Environments

There are 5 software environments to replace the OPEN-R application layer and program the AIBO. A short introduction of each software environment will be presented below.

OPEN-R SDK

Sony developed an OPEN-R SDK with which applications that make use of OPEN-R system level functions can be developed. The OPEN-R SDK is a cross development environment based on gcc

(C++). Applications created with the OPEN-R SDK can run on AIBO, but the application has to be copied to a programmable memory stick first. The OPEN-R SDK can be considered the most basic of the packages. In addition, the OPEN-R SDK contains "Remote Processing OPEN-R" (RP_OPEN-R). RP_OPEN-R is a remote processing environment where OPEN-R based programs can be executed on machines other then AIBO [51].

URBI

URBI, Universal Robotic Body Interface, is a scripted command language used to control robots (AIBO, pioneer,etc). It is a robot-independent API based on client/server architecture. In the OPEN-R programming model the URBI server can be viewed as another object. The developer can make use of the URBI server in two ways: via a computer through the wireless LAN using the liburbi C++ (external client) or through direct inter process communication using liburbi-openr (onboard client) [51].

Tekkotsu

Tekkotsu is a framework built on top of OPEN-R SDK. This means, that in order to use Tekkotsu, the OPEN-R SDK also has to be installed. Tekkotsu offers a way to interface with WLAN. Joints, headmovement camera etc. can be controlled via wireless LAN. The programming model with URBI also holds for Tekkotsu. (Tekkotsu server is also an object running on OPEN-R).

The advantage of Tekkotsu is that it offers higher level commands (instead of moving individual joints, one can issue commands like “walk”). Furthermore the Tekkotsu framework aids people who develop objects intended to work on the AIBO (with Tekkotsu) by adding a level of abstraction. So instead of having the to know the message passing details of ones object with other objects (like in the URBI) a Tekkotsu programmer can think in terms of behaviors [51].

OPEN-R framework

AIBO Remote Framework is a Windows PC application development environment based on Visual C++ with which you can make software that works on a Windows PC. The software can control AIBO (ERS-7) via a wireless LAN. Multiple AIBO and PC applications can be connected at the same time. OPEN-R framework also offers high level commands [51].

R-Code

R-CODE is a high-level, interpreted script language created for AIBO Master Studio. It is intended for hobbyists and end-users. An R-CODE script is interpreted and executed by the R-CODE interpreter object, which is actually an OPEN-R object. As R-CODE is an interpreted script, it is not suitable for computing-intensive processing [51].

4.3. Working of the AIBO Watchdog

The AIBO watchdog can be viewed as an agent. It perceives its environment through sensors and acting upon that environment through actuators. A global description of this process will be explained in this section. Figure 4.5 shows a simple representation of this process.

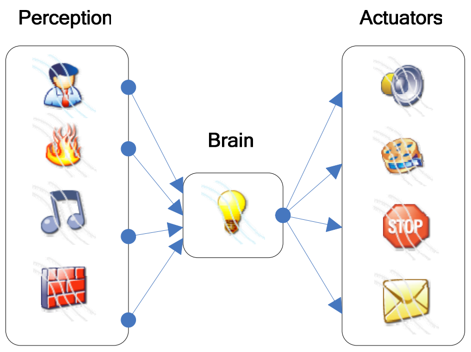


Figure 4.5: A simple representation of the working of AIBO watchdog.

4.3.1. AIBO Perception

The perception of the AIBO relies on its input sensors. Every sensor can sense some features of an object. Features of an object that an image sensor can perceive are for example the color and shape of an object. The sound frequency is an example of a sound feature that the sound sensor can perceive. Figure 4.5 shows some possible perception by AIBO watchdog. Objects can have sound features and image features or only one of them. Other features are also possible. The big question is whether they are detectable by the available sensors. That is the reason why some sensors are more important than the others in certain environments.

4.3.2. AIBO Brain

As mentioned before AIBO lacks of a good reasoning system for the AIBO watchdog. Therefore an intelligent reasoning system needs to be developed. To have an intelligent reasoning system the perceived features from its sensors have to be combined and synchronized on time. As a result a clear picture can be drawn about the situation and the most appropriate reaction can be derived.

Prior to being able to perceive features of objects a complete environment needs to be designed. The relevant objects for the AIBO watchdog, which the reasoning system will reason about, need to be described and modeled. This world model will contain the object list of the AIBO environment and a description about the transformation of objects into features.

In order to get new inputs from its environment the AIBO watchdog needs to be able to walk around and perceive information from the new location. A route planning system is therefore a part of the AIBO brain. Depending on the situation this planning system gives the safest, fastest or shortest path or a combination of these aspects.

4.3.3. AIBO Actuators

The actions that the reasoning system prescribed need to be executed by AIBO. The actuators of AIBO, e.g. legs, head, wireless connection, speaker, can work in parallel or sequential. Advanced and efficient movements can be executed, but the synchronization aspect plays an important role to coordinate the AIBO successfully. Figure 4.5 shows some possible actuators by the AIBO watchdog.

4.4. Functionalities

In this section the functionalities of the AIBO watchdog will be explained. Based on the hardware capabilities and limitations the functionalities of AIBO can be determined. The functionalities of the AIBO watchdog can be divided into 4 categories: basic physical operations, basic technical operations, advanced operations, high level operations.

1. The basic physical operations deal with physical joint movements of the AIBO. These operations form the base for all functionalities of the AIBO.
2. The basic technical operations deal with the technical devices in the AIBO, such as sound and display devices.
3. Advanced operations combine the basic physical operations and basic technical operations to realize a complex action.
4. After being able to execute advanced tasks by the AIBO, high level operations provide the owner the one-touch command functionality, e.g. by touching its head it can patrol its environment autonomously and reports the owner when something suspicious has been found.

4.4.1. Basic Physical Operations

Leg Movements

- Walk slowly.
- Walk normally.
- Walk fast.
- Turn left.
- Turn right.
- Stand still.
- Lay down.

The pace of walking should be fully dependent on the environment and the state of the AIBO. In emergency AIBO will walk faster than usual. When there are obstacles on the path, AIBO will turn left or right to avoid the obstacle and the pace of walking should also be adapted.

Head rotations

- Rotate head to left.
- Rotate head to right.

Rotating the head is necessary for detecting obstacles on the path. By rotating the head to left and right, the AIBO is able to detect obstacles on its left side and right side.

4.4.2. Basic Technical Operations

➢ Input operations

Taking picture

- Capture the scene.

When something suspicious or abnormalities are found, the AIBO can capture the scene and send it by email. The owner can have a better look on the situation. Beside of that this functionality can also be used for recognition of objects.

Detecting sound sources

- Amplitude and frequency of sound.
- Direction of sound source.

When the AIBO watchdog hears some sounds, it will determine the amplitude and frequency. These data are needed for classifying the sound source. When possible the direction of the sound source needs also to be estimated.

Measuring distance to obstacle

- Measure distance to obstacle with Infrared sensor

By using the Infrared sensor on the head of AIBO, the distance can be measured to the obstacle. Collision can beforehand be avoided.

Touching objects

- Touched objects with touch sensor.

When the touch sensors bumped onto an obstacle, the AIBO have to react on it. Sometimes it is also necessary to notice when it do not bump onto an object. This is the case when one of the legs of AIBO does not touch the ground, after a full step movement of the AIBO. This situation can indicate that AIBO is standing on 3 legs and it is going to fall down.

➢ Output operations

Playing sounds

- Play a wave/midi file.

By playing sounds AIBO can alarm the people around the AIBO watchdog. Normally when the AIBO have not encountered some suspicious things, it can play a tune to let the people know that everything is safe. When it encounters some abnormalities, it can play another pre-programmed tune. It is a one way communication with the people. This functionality can be switched off when there is nobody at home.

Displaying mood on its face

- Display the LED’s on the face.

By displaying the LED’s on the face of the AIBO, the people will know what the AIBO watchdog is doing right now. The LED’s can also show whether something serious has happened during the last hour. For example a green circle on its face indicates nothing serious has happened in the last hour and a red circle when something serious has happened.

Sending emails

- Send emails to the owner.

When the owner is not at home, this is the only communication channel with its owner. Serious event can be reported to the owner.

4.4.3. Advanced Operations

Recognizing objects

- Color, shape and material.

By capturing the images by the camera of the AIBO, an analysis of the color, shape and material of the objects in the picture can be made. Using predefined suspicious color, shape and material a match can be made with the captured picture.

Scanning area

- Obstacles on path.

By scanning the area with camera and Infrared distance sensor the AIBO is able to detect obstacles on its path.

Obstacles avoidance

- Bumping on the obstacle.
- Avoiding collision with the obstacle.

When the AIBO is not quick enough to detect the obstacle, it will bump onto the obstacle and choose another path to walk. Choosing another path is necessary to avoid bumping onto it forever. When the AIBO is quick enough to detect the obstacle, AIBO will choose another path to walk.

Moving to another room

- Locate its current position and find the destination.

The AIBO must be able to locate its current position. This can be done by tagging each room or finding key objects. Using this information and a map the AIBO is able to move to another room.

Detecting abnormalities

- Pre-programmed suspicious color or shape.
- Missing objects.

After pre-setting the suspicious color and shape, every time when AIBO encounters that color and shape, the AIBO will alarm itself and the owner that something suspicious has been found. Based on the kind of abnormality an appropriate action will be carried out. In a predefined world environment, AIBO is able to detect missing objects and changes in the predefined world environment. An appropriate action can be linked to the kind of change.

Moving to a sound source

- Sounds amplitude and sound frequency.
- Sound direction.

When the AIBO hears some sounds, it will move to the sound source with the highest sounds amplitude and thereafter searching for the highest frequency. The sound direction of the sound source has to be estimated.

4.4.4. High Level Operations

Patrolling the environment

- Walking around the whole house.

The AIBO is able to patrol the whole environment. The AIBO must be aware of its current location and be able to move from one location to another location in the home environment. It must not be stuck in one room and it must have the intelligence to escape.

Finding suspicious events

- Detect abnormalities.
- Detect sound.

When AIBO detects sounds or abnormalities, it will explore the environment and find the cause of the abnormalities. Thereafter an appropriate action will be carried out.

5

Model Design

"Imagination is the beginning of creation. You imagine what you desire, you will what you imagine and at last you create what you will."
George Bernard Shaw

This chapter presents the models regarding the design of the AIBO watchdog. First the requirements analysis of the AIBO watchdog will be discussed; thereafter the developed architecture and finally the used models will be presented. Furthermore the global interaction process of AIBO with its environment will be explained. Thereafter an architecture will be presented based on this interaction process. Additionally a navigation method will be presented including the test environment.

5.1. Requirements Analysis

In this section the requirements of the ideal AIBO watchdog will be discussed and the scope of this project will be determined. There are three kinds of requirements: functional requirements, nonfunctional requirements and pseudo requirements. Functional requirements describe the interactions between the system and its environment independent of its implementation. The environment includes the user and any other external system with which the system interacts. On the other hand nonfunctional requirements describe the user-visible aspects of the system that are not directly related with the functional behavior of the system. Nonfunctional requirements include quantitative constraints, such as response time or accuracy. Furthermore pseudo requirements are requirements imposed by the client that restrict the implementation of the system.

5.1.1. Functional Requirements

- *Ability to move freely in its environment*

The first requirement of the AIBO watchdog is that it must be able to move freely in its environment, a home environment. This implicates that it can avoid obstacles and avoids falling down from the stairs. Using the distance sensor and edge sensor of the AIBO these 2 situations can be solved. Obstacles which are not directly blocking the AIBO watchdog need also be avoided. A good example of such kind of object is a small chair with 4 legs. At the first sight it seems that the AIBO can walk through the hole, but when AIBO is walking through it, AIBO head may collide with the upper part of the chair. Therefore AIBO must be able to move freely its environment without damaging itself.

- *Creating a map of the environment*

When AIBO has started up, it can perceive the environment with its sensors. At this stage AIBO can create a map of its environment. So by exploring its environment a real-time map can be created. This approach requires at least 2 cameras to estimate the distance to this object. The built-in distance sensor can give a good support for this approach. But this approach heavily relies on its vision. Therefore a good lighting condition is required. A less flexible approach is importing the map of the environment in the AIBO. This can save a lot of time for the AIBO and it can patrol the environment immediately. If AIBO discovers some changes in the environment during patrolling, the imported map will be updated. As a result AIBO will always have an updated map even when the imported map was not up to date.

- *Navigation in the environment*

Based on the created map and the features around the AIBO, AIBO must be able to locate itself on the map. After knowing the goal position AIBO has to calculate the best route to the goal position. The best route can be based on the shortest path, the fastest path or something else. The navigation algorithm can be based on a discrete representation of the map or the continuous representation. Using the discrete approach it is much easier to navigate to the destination.

An example of the discrete approach is using waypoints. We divide the whole environment into equally divided squares, so called cells. The start location is the middle of the cell and the end location is the middle of an adjacent cell. An adjacent cell can be a cell which is vertically, horizontally or diagonally related to the start cell. In a continuous approach AIBO will update its location after every small step. This approach is more natural and flexible. The drawbacks of this approach are that the navigation algorithm is very complicated and it requires a robot which is able to move accurately.

- *Objects recognition*

When AIBO is able to move around freely, AIBO needs to react to objects that it encounters. Therefore recognizing the objects is the next step. In a real environment it is very difficult with recognizing the objects, especially when it is not possible to define the objects unambiguously. A simple example is defining the features of a chair. It can have 1 leg, 2 legs, 3 legs or 4 legs. If it has 4 legs, this can also be a table. To be able to recognize objects properly, we need a classifier which is able to classify the objects properly based on its features.

➢ *Objects tracking*

It is not sufficient only to recognize the objects. Moving objects need to be tracked after recognizing. It is important to have a motion estimation algorithm to find the moving object in the next frame of the incoming video. When AIBO needs to keep track of an object while AIBO is walking, this motion estimation algorithm has to offer assistance.

➢ *Scenario reasoning*

AIBO should have some sense of context awareness. After discovering certain objects AIBO should react to it. If it was a normal object, nothing special is needed to carry out. Otherwise the discovered object can indicate that a certain scenario has already started. Therefore a good reasoning system is needed to reason about which scenario has started based on the incoming objects during patrolling.

➢ *Prioritizing the scenarios and remembering the scenarios*

After knowing which scenario is going on it does not mean that scenario need to be solved. Two or more scenarios can occur at the same time. The reasoning system needs to be able to prioritize the occurring scenarios. Scenario which has a higher priority needs to be carried out first and the rest of the scenarios will be solved afterwards. Therefore the AIBO also needs to remember the scenarios that have occurred till now.

➢ *Remembering the useful objects*

During patrolling AIBO can encounter objects that can be used to solve scenarios. When AIBO needs to escape its environment, it is necessary that AIBO knows the location of the door. If it has encountered the door before, it is useful to put that in its memory. As a result the intelligence of the AIBO can be increased.

➢ *Assigning Actions*

After knowing the occurring scenarios AIBO must react on it, or in other words AIBO must try to solve the scenario by itself or with help of others. The reactions that AIBO needs to carry out can be given beforehand to the AIBO or let the AIBO learn the reactions by training. By implementing the actions for every scenario in the AIBO, AIBO can react on the predefined scenarios properly. The advantage of this approach is that it can be used immediately after adding the programming codes. Adding new actions and removing actions can very easily be done. While training is time consumptive and it is less flexible with adding and removing actions. But when there are a lot of scenarios and possible actions this approach can produce the output much faster and less code is needed.

➢ *Working autonomously*

The ideal AIBO watchdog should be able to move on its own without help of other tools or human being. So the complete program to control the AIBO watchdog should be implemented in the memory of AIBO. AIBO uses a special Sony memory stick to load the program in the working memory. The capacity of this memory stick is at most 16MB. Therefore if we want to create an

AIBO watchdog, the whole program can be at most 16MB. Since object recognition programs are very large and processor intensive a solution can be running the software on an external PC and control the AIBO watchdog by this external PC. This approach creates a non-autonomous AIBO watchdog which relies on the robustness of the external PC and wireless communication.

➢ *Automatic charging*

Activation of a scenario should not only consider external incoming events, but also internal events. Using one full charge of the battery AIBO is able to move around for 2 hours. Therefore it is necessary to charge the AIBO a few times a day. If there are no people present for a long period to charge the AIBO, the AIBO watchdog concept is useless. Therefore AIBO watchdog must be able to charge itself, when its battery level is very low. The Sony software that has been included with the AIBO robot already has this functionality. It proves that automatic charging is not just a concept, but it does work in practical.

5.1.2. Nonfunctional Requirements

➢ *Real time patrolling*

The AIBO watchdog needs to be able to react on the objects as quick as possible. Therefore it is required that AIBO watchdog is able to patrol the environment in real time and react on events within 1 second.

➢ *Loudness of alarm sounds*

When AIBO is creating an alarm sound to warn the surrounding people, the sound must be as loud as possible but not damaging the ears of people. Human beings have different sensitivities for different sound frequencies. The spectrum of the frequencies that human beings can sense is between 20 Hz and 20000 Hz. According to the Fletcher - Munson curve [53] the most sensitive spectrum is between 100 Hz and 10000 Hz and the corresponding sound pressure level is between 50 and 70 dB. Therefore these numbers will be the sound spectrum for the alarm sounds.

➢ *Quality of captured pictures*

For recognizing the objects in pictures by people and image processing software it is necessary to have a picture as large as possible. The largest resolution of the pictures is only 208x160 pixels due to the camera limitations. Therefore the resolution of the captured images has to be 208x160 pixels and saved in jpeg format.

5.1.3. Pseudo Requirements

➢ *Modular and flexible*

The requirements that are mentioned in this chapter, each of them can be achieved with several approaches. Due to the time aspect it is not possible to choose the best option at each requirement. Therefore the architecture of the AIBO watchdog must be designed modular, so changes in future work will not lead to implementing the AIBO watchdog from scratch on. But only by replacing a component is sufficient. The flexibility of the program is also an important issue. Not everyone likes the barking sound that is standard implemented in the program. So it must remain flexible and provides the ability to be adapted to everyone's flavor.

5.1.4. Scope of This Project

Every requirement of the ideal AIBO watchdog consist problems that need to be solved. Due to the time constraint we are not able to solve them all, but we need most of the requirements to show the correct working of the AIBO watchdog. Therefore we will fulfill most of the requirements and sometimes the simplest approach will be chosen. The list below will discuss to which extent the requirements will be fulfilled:

- **Ability to move freely in its environment**
  AIBO watchdog is able to avoid collision with obstacles while patrolling.
- **Creating a map of the environment**
  AIBO knows its environment after initialization.
- **Navigation in the environment**
  AIBO can calculate the most appropriate route and navigate to a certain point.
- **Objects recognition**
  Recognition of objects will be completed by simulation. The incoming features of objects will be simulated.
- **Objects tracking**
  No moving objects are taken into account.
- **Scenario reasoning**
  AIBO watchdog is able to reason the scenario of the current situation.
- **Prioritizing the scenarios and remembering the scenarios**
  AIBO knows what to do when more than 1 possible scenario occurs.
- **Remembering the useful objects**
  AIBO understands that some objects can be used and when to use it.
- **Assigning actions**
  AIBO knows what kinds of actions are needed to be executed in a certain situation.
- **Working autonomously**
  AIBO is controlled by an external pc
- **Automatic charging**
  The user has to charge the AIBO manually.
- **Real time patrolling**
  AIBO reacts on real time objects within 3 seconds.
- **Loudness of alarm sounds**
  Frequency of alarms sounds will be between 100 Hz and 10000 Hz and sound pressure level is between 50 and 70 dB.
- **Quality of captured pictures**
  Resolution of captured pictures is 208x160 pixels in jpeg format.

- **Modular and flexible**
  The AIBO watchdog architecture is modular and flexible.
- **Software**
  To continue the work of the predecessors the AIBO watchdog will be implemented in Java and Jess.

5.2. Interaction Process with Environment

Understanding the interaction process of the AIBO watchdog is crucial to develop the reasoning architecture of the AIBO watchdog. Figure 5.1 illustrates this interaction process.

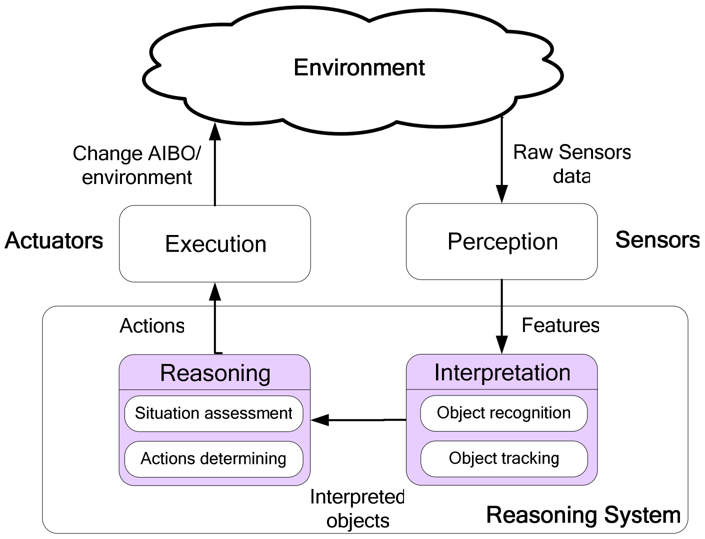


Figure 5.1: The global interaction process of AIBO and its environment.

The environment contains a collection of physical objects which are spread across the environments area, e.g. table, door or fire. All objects have features which can be perceived by the AIBO sensors. The AIBO robot dog has 4 types of sensors which can interact with its environment: sound sensor, image sensor, distance sensor and touch sensor. The inputs of these sensors create a feature list at a certain time and location.

The features in the feature list will be used for recognition of the objects in the environment. The correctness of this recognition is heavily depending on the correctness of the sensor’s signal and the quantity of features that the sensors can distinguish. Furthermore due to ambiguity of the features the interpreted objects will not always match the original object. This problem can partly be solved by introducing more detectable features to decrease the ambiguity level. The interpretation process will result to the interpreted objects. Furthermore to interpret moving objects it is necessary to track the objects during a certain time period. Tracking can be achieved by comparing the current video frame with the previous ones.

The interpreted objects will be presented to the reasoning process for understanding the current situation and deciding the actions that need to be carried out. After executing all actions the environment will provide the sensors with new data and this process will continue eternally. The 4 interaction process will be explained in detail in the following subchapters.

5.2.1. Environment

The environment contains physical objects that the sensors of AIBO partially can perceive. These physical objects can be moving objects or nonmoving objects. The common objects that do not need to be reacted to by the AIBO watchdog are called the normal objects. Other the other hand objects that require special execution by the AIBO watchdog are called the special objects. These special objects can be divided into 2 categories: causal and result. The causal objects will cause the presence of other special objects and result objects on the other hand are the resulted special objects caused by one or more causal objects. Therefore it is necessary to find the cause of these resulted special objects to understand the situation. Figure 5.2 shows some examples of the normal and special objects.

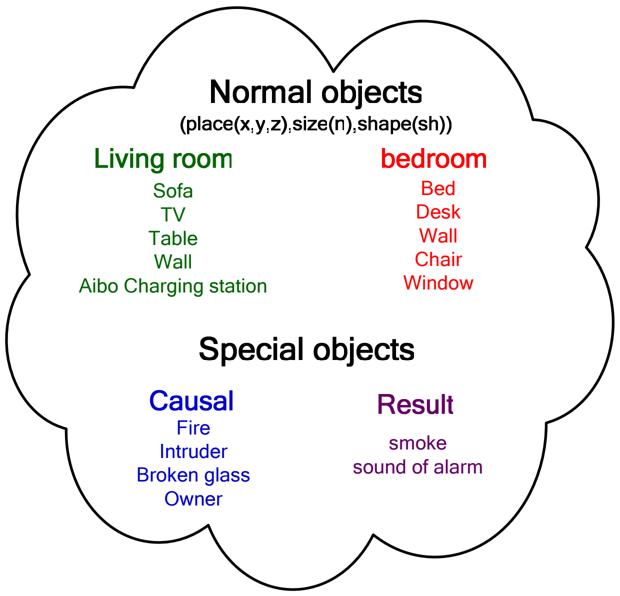


Figure 5.2: Example of the content of the environment which will be used in the simulation process.

As mentioned in the requirements analysis section the information about the AIBO watchdog environment will be imported in AIBO during initialization. The normal objects will be imported in AIBO watchdog during initialization. On the other hand to be able to show the correct reasoning of the special objects the special objects will not be imported in AIBO watchdog prototype during initialization, but AIBO watchdog need to discover them during patrolling.

5.2.2. Perception by Sensors

The sensors raw data of the perceived objects in the environment will be preprocessed by the preprocessing programs. These programs will filter out the noises and irrelevant redundant data and process some useful outputs. These outputs are the features of the objects. Figure 5.3 illustrate these processes.

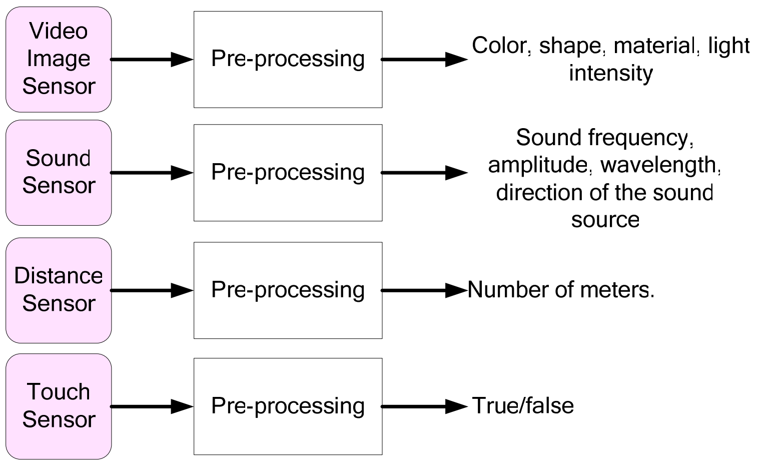


Figure 5.3: The pre-processing diagram of the sensor's input and output data.

The resulted outputs of the preprocessing process need to be transferred to the next stage of the interaction process, interpretation process. Figure 5.4 shows an example of the data transferring to the next stage.

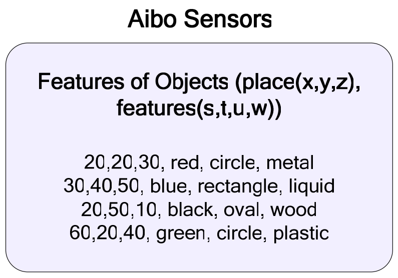

Aibo Sensors
Features of Objects (place(x,y,z), features(s,t,u,w))
20,20,30, red, circle, metal
30,40,50, blue, rectangle, liquid
20,50,10, black, oval, wood
60,20,40, green, circle, plastic

Figure 5.4: Example output of perception process transferring to the interpretation process.

In this example only the features color, shape and material are shown. There are much more features possible, such as sound features. The more features you have and the more accurately you can describe, the less ambiguity is left and the more accurately AIBO can recognize.

5.2.3. Interpretation

The interpretation process consists of 2 main processes: object recognition and object tracking. The received features from the perception process need to be processed by the object recognition process. Based on the incoming features objects can be recognized. The accuracy of the recognition process depends on the ambiguity of the perceived features. More classification classes can reduce this problem. For monitoring the moving objects it is also necessary to track the movement of these objects. The recognized objects and eventually its movements will be transferred to the next stage. Figure 5.5 shows an example how the interpretation process works.

This example shows three features of the object at a certain location and concludes that a human being has been detected. But ambiguity still exists with these kinds of rules. The conclusion did not tell us whether it is a living human body or a corpse. If the AIBO has detected the first possibility, then it does not need to execute any actions. But if AIBO has detected the last one, it should carry out some appropriate actions, e.g. take a picture of the corpse, send the picture to the police and alarm people around it. Because the real world is an open system and changes over time, it is not possible to describe the world with limited rules.

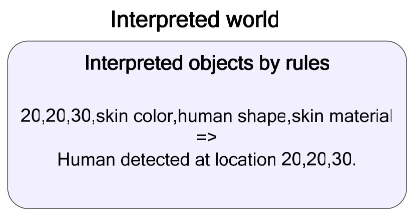

Interpreted world
Interpreted objects by rules
20,20,30,skin color,human shape,skin material
=>
Human detected at location 20,20,30.

Figure 5.5: Example of the interpretation process.

5.2.4. Reasoning Process

The reasoning process consists of 2 main processes: situation assessment and action determining process. After receiving the information about the recognized object, such as position, orientation, state and type, the current situation needs to be understood. By reasoning the possible situations that the current objects can lead an appropriate plan need to be developed to solve the current situation. It is important that the time aspect will also be taken into account during reasoning. Since time is also a property of objects [17]. The solution is the actions that AIBO needs to carry out. These actions will be transferred to the next process, execution process.

An object can be at position (x,y,z) at a certain moment, but it does not implicates that the object will be at the same position after a certain time period. There are three reasons why the position of the object has changed.

1. People moved away the object.
2. Object moved away by itself.
3. Object has been transformed.

The last possibility describes that objects can have a different shape, color material and other features during interaction with its environment. A good example is an ice cream in the open air during a hot summer day. Depending on the time period that the ice cream is exposed in the open air environment its material, shape and color will be changed. The degree of its changing is dependent on the time aspect [18].

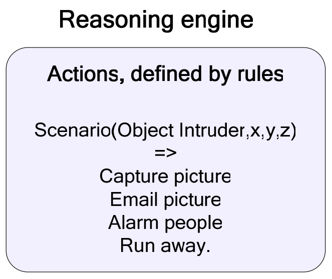


Figure 5.6: Example of rules in reasoning engine.

A normal way to deal with the time aspect is introducing a memory component in the reasoning engine. Correlation between the ice cream before melting and melted ice cream can be understood by the reasoning engine. The memory component is the key component for human beings to be able to think and reason. It is a part of our intelligence [21]. Figure 5.6 shows an example of a reaction on the recognized object.

5.2.5. Execution Process

The received actions from the reasoning process will be stored in a queue. Two or more consecutive actions that can be executed in parallel will be grouped and executed in parallel. Otherwise these actions will be executed in the serial way, one by one. By these actions AIBO and its environment are changed. The new sensors inputs will lead to new actions. The interaction process of the AIBO watchdog with its environment will continue eternally.

5.3. Architecture Reasoning

After understanding the global interaction process of the AIBO watchdog with its environment the architecture developing process can start. The total architecture should be developed to match the interaction process. Figure 5.7 shows the developed architecture of the AIBO watchdog. The main focus of this project is the reasoning system and partly of the execution process to show the correctness of the reasoning process.

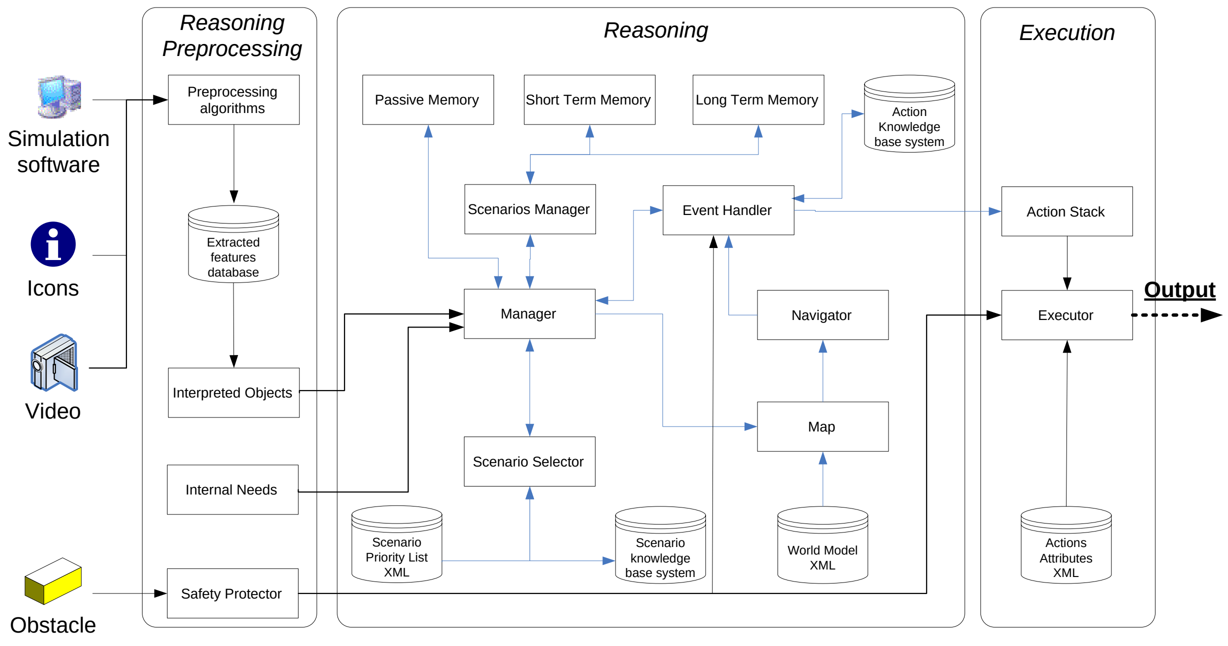


Figure 5.7: The designed architecture of the reasoning component of AIBO watchdog.

Only the reasoning process has been totally worked out. The execution process is modeled to the level that it can cooperate with the reasoning system. All components of this architecture will be explained in details in the next subchapters.

5.3.1. Reasoning Preprocessing

This part of the architecture deals with the data preparation for the reasoning process. First the data from sensors will be preprocessed with software and the extracted features of objects will be stored in a features database. The interpretation process will recognize the objects from the features database. These recognized objects will be passed to the manager of the reasoning system.

- **Preprocessing algorithms**

This component will receive sensors input of the environment. Environment can be simulated by a program which will give the objects to the preprocessing component automatically when it reaches a certain place. In this situation the tasks of the preprocessing component is very simple. According to a predefined look-up table the features of that object can be found and these data will be transferred to the features database.

Another approach for representing the environment is with icons in a real environment. Instead of real objects it will show an icon of the real object. The icon represents the real object. In this case the preprocessing component contains a software program which can recognize the icons. Usually this kind of programs contains internally a features database of all possible icons. The external features database will be replaced by the internal one. The output data can be transferred to the interpreted object component directly.

The most difficult approach for representing the world environment is the real world itself. The software in the preprocessing component has to recognize real objects in the real world environment. This can be completed by finding the shape, color and sound of the objects. All kind of features of objects can be used, e.g. material and shadow. After extracting all these features from its sensors these data will be transferred to the features database for further matching. This is the ultimate goal of the AIBO watchdog project an autonomous robot which is able to navigate in a real world and able to understand the real objects in the environment as the human beings. Before the developed architecture can achieve this stage, the developed reasoning system firstly has to be able to conquer the first 2 approaches.

- **Extracted features database**

The extracted features database contains the result of the preprocessing programs. The structure of the features database is fully dependent on the preprocessing programs. Figure 5.8 illustrates an example of the structure of the features database.

- **Interpreted Objects**

The Interpreted Objects component tries to use the content in the extracted features database to recognize the original object. The structure of this component is fully dependent on the previous two stages. Many approaches can be used for this component. A neural network approach can be used to classify the features in the features database. If the possible objects are very limited and the uncertainty is very low one can also consider an expert system approach for this component. The results of this component are objects. These objects will be given to the manager component in the reasoning system.

These three components in the reasoning preprocessing can be merged into one program. But the tasks of these three components will still be recognizable in the merged program.

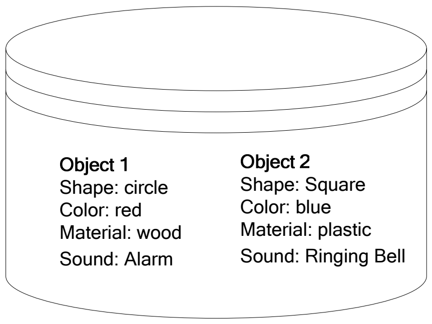


Figure 5.8: Example structure and contents of the extracted features database.

- **Internal Needs**

The internal needs component takes care of the AIBO health and mood. AIBO was designed to accompany people who feel lonely. By adding this component AIBO can also express its desires and needs. The complexity of these desires needs to be modeled in this component. Another important factor in this component is it desires for food when it is hungry. When the battery is running low, the AIBO watchdog needs to find the charging station and charges itself up. The low battery level message will be sent to the manager component which will handle it professionally.

- **Safety protector**

The safety protector component monitors the input signals from the touch sensors of the paws and the distance sensor. This component is added in the architecture to protect the AIBO from damages. When the input signals of these sensors differ from the values in the safety protector component, the execution of the actions needs to be terminated immediately and the executed physical actions need to be reversed. The event handler component will find the last executed actions and a reversal of these actions will be made. The new path needs to be recalculated by the Navigator component to avoid the same dangerous situation.

5.3.2. Reasoning

The reasoning process contains a few components which deal with the proper working of the reasoning system. The results of the reasoning process are the actions that need to be executed. These actions will be put on the action stack. This subchapter will explain the details of the reasoning process.

➢ Manager

The manager component manages the working of the first part of the reasoning process. The manager only knows the sequential actions that are needed to be executed. It takes care of the execution plan and this is its limited functionality. It acts like a middleman. It gets information from a component and passes it on to another component. When all two-ways linked components, except the event handler component, have communicated with the manager component the end result, a scenario, will be given to the event handler component to decide the actions. Figure 5.9 and Table 5.1 illustrates the working of the manager component.

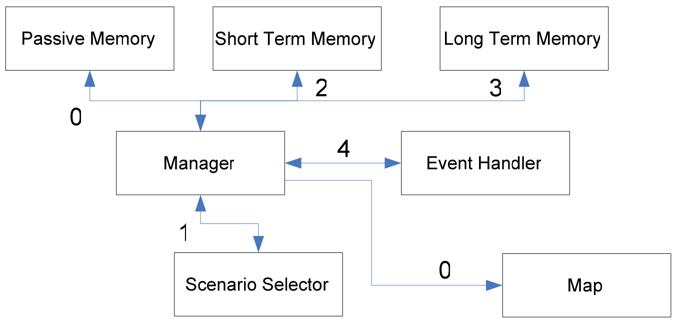


Figure 5.9: Communication sequences of the manager component. First recognized objects will be stored in passive memory and map component. Thereafter a scenario is selected and saved in the short term and long term memory. The most important scenario is passed to the event handler.

➢ Passive memory

Human beings can memorize objects unaware that they have seen. When the time comes that they need it, they will try to remember what it exactly was and where did they exactly encounter it. This is a kind of passive memory that human beings normally are not aware that they use it, but it will sometimes be activated when they need that information in certain situations. This kind of memory has been mentioned in chapter 2, the episodic memory. Therefore this kind of memory will be also included in the design of the AIBO watchdog to make it more intelligent. Figure 5.10 illustrates this concept.

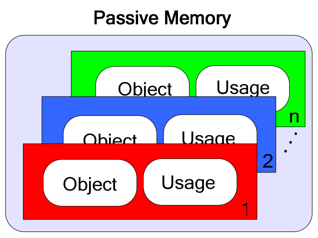


Figure 5.10: The content and structure of the passive memory component. The memory contains a vector of the name and coordinates of objects and also its usage.

The passive memory in the AIBO watchdog will memorize the objects that it has perceived and its usage. For example when AIBO has encountered a door, then AIBO will assign a meaning to this object. A door can be used to escape from this environment when necessary. The more useful objects that AIBO has perceived during its patrolling, the more useful reactions AIBO can make, when it encounters some suspicious situations.

Table 5.1: Sequence of the working of the manager component.

Sequence	Working
0	The incoming objects need conditionally to be stored in the passive memory for later usage. These objects need also to be placed on the map. This way the map will also be updated. These 2 actions need to be done before the manager component communicates with the other components. But the sequences of these 2 actions are independent of each other.
1	Manager component communicates with the scenario selector component to determine the new scenario based on the incoming objects. The new objects will be communicated to the scenario selector and the scenario selector will provide the manager the changes of the current scenario or a new scenario.
2	Manager component checks whether the new scenario has a higher priority than the scenario in short term memory and try to put it in short term memory.
3	Manager component could not save the new scenario in the short term memory, because the current scenario in short term memory has a higher priority. Therefore the new scenario will be put in the long term memory. This scenario will temporarily be ignored.
4	Manager component tells the event handler component whether the current scenario has changed or not based on the incoming objects. If it has changed, the new scenario will be communicated to the event handler component.

Every time when AIBO watchdog encounters a special object, it will search in its passive memory to find a useful solution. When there are useful objects found, AIBO will set its new goal point to that location. Otherwise a default routine which is pre-programmed will be executed. The size of the passive memory is infinite, but usually it is limited by the boundary of the computer where the program is running.

New incoming objects which are identical to the objects in the passive memory will replace the old objects. Figure 5.11 illustrates the working of this concept. This approach is chosen to ascertain the newest solution is the closest solution to solve the current situation. This last-in-first-out approach will have the same handicap as the short and long term memory concept. When walking in a round and only a few useful objects are discovered this approach will not give the best solution. But in general, considering the complexity of the evaluation algorithm, this approach should give the AIBO watchdog the best behavior. A better algorithm can be a heuristic function which will take current AIBO position, goal position and estimated walk distance into account.

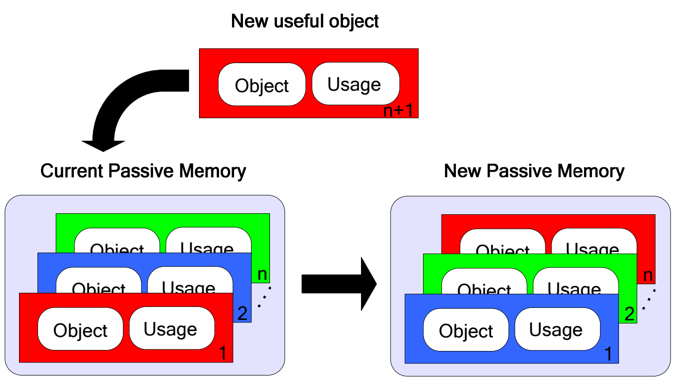


Figure 5.11: The adding process of a new useful object. When a new useful object has arrived which has already been detected in the past, the new object will be put on the top of the stack.

➢ Scenario knowledge base system

The scenario knowledge base system contains a lot of possible scenarios that can occur in a home environment. These scenarios are described using a card system. All these cards are sorted in a card-tray. Figure 5.12 shows an example of this card-tray system.

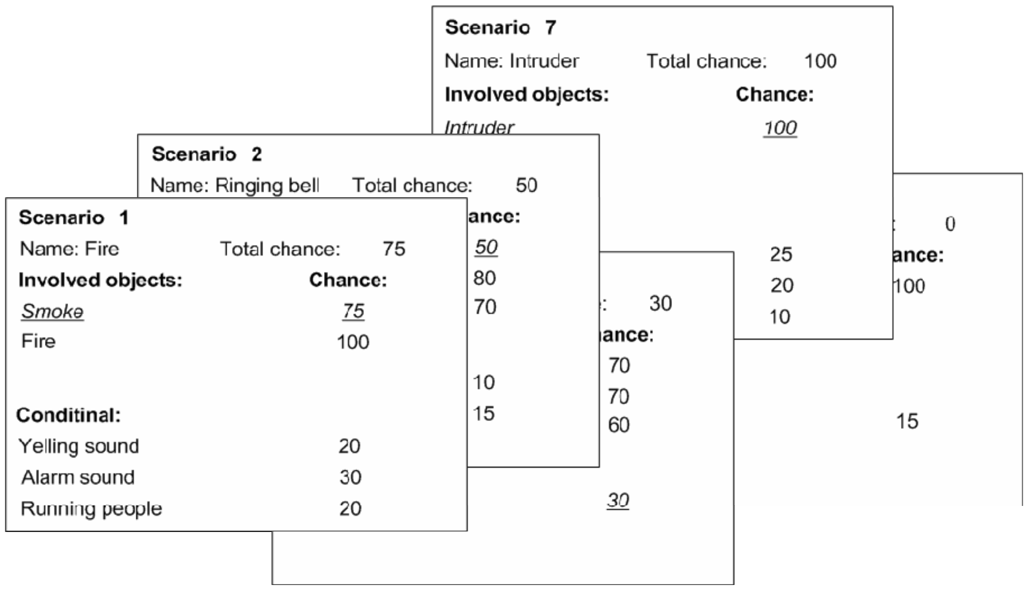


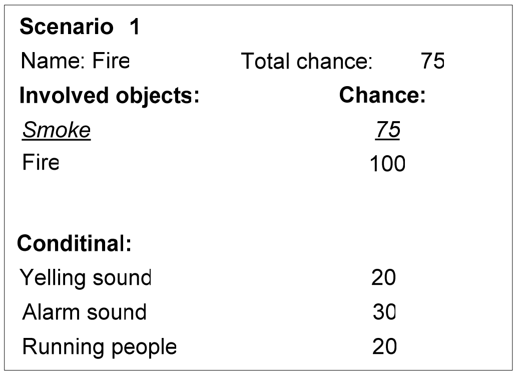
Figure 5.12: Result of a card tray system in a scenario knowledge base system with a threshold of 50. The scenarios which have a value larger than the threshold and at least one involved object has been detected will be selected.

Every time when an object arrives, the object will be matched with all objects in the card tray. For example when the new object smoke has been detected, the state of all scenarios which contains the smoke object will be updated. The object smoke will be highlighted and the total chance of that scenario will be updated. Figure 5.12 shows a situation when the threshold has been set to 50. Every scenario which has a chance probability of 50 or higher will be selected, but this is only the case when there is an involved object highlighted. Otherwise that card will not be selected as a potential scenario. So even when all conditional objects has been highlighted and the total chance is more than the threshold, that card will not be selected if there are no involved object highlighted.

When an incoming object changed a value of a card, but it was not sufficient to trigger the scenario to be selected, this change will be reported to the manager to carry out a small sub-action. A small sub-action can be take a picture or capture the sound. These sub-actions do not interfere with the main actions and can be executed in parallel. The idea behind this concept is that the new low-prioritized scenario will not be totally ignored, but it will be partly executed.

➢ Designing a scenario

If one has written down all involved objects in a scenario, how can one assign chances to these objects? Firstly there is a threshold defined, e.g. 50. Secondly split the involved objects into conditional involved objects, objects which can occur in a scenario, but not necessary, and involved objects which certainly will occur in this scenario. The assigned chances to the conditional involved objects are not the most important, therefore based on your own intuition the values can be assigned, but this has to be assigned relatively to the other involved objects. The last part is assigning the chances to the involved objects. We have to consider which of the involved objects can ascertain the existence of this scenario. These involved objects will have a chance of more than the threshold. As a result this scenario will be ready to be selected. The other involved objects will have a chance below the threshold. If combining these involved objects can lead to certainty of the scenario, then the sum of the chances of these objects will be more than the threshold. Otherwise the sum of the chances of these objects is lower than the threshold. Figure 5.13 illustrates the result of this process.



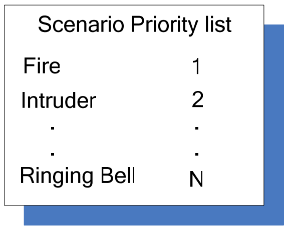
Scenario 1		
Name: Fire	Total chance:	75
Involved objects:	**Chance:**	
Smoke	*75*	
Fire	100	
Conditinal:		
Yelling sound	20	
Alarm sound	30	
Running people	20	

Figure 5.13: Result of the chance assigning process to the scenario objects. Scenario 1 has a chance of 75, because of the detected smoke object.

Smoke is necessary in a fire scenario, because when smoke exists, there must be something burning. Fire object is the core of the whole scenario. Therefore the maximum chance has been assigned to this object. The conditional objects alarm sound will not always exist in a fire scenario, because it will only be heard when someone has hit the alarm button. Yelling sound and running people can only be the case, when there are people in the environment who have detected the fire scenario.

➢ Scenario Priority List - XML

This scenario priority list deals with the priority of the occurring scenarios. When there are more than 1 scenario occurs at the same time, the scenario selector will select the most important scenario based on the scenario priority list. This scenario will be executed first. Figure 5.14 shows an example content of a scenario priority list.



Scenario Priority list

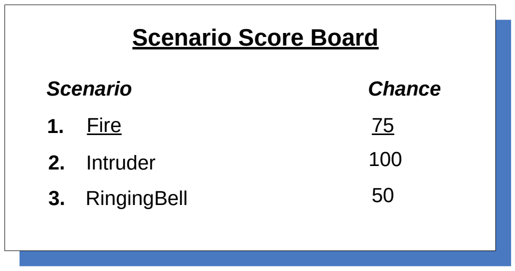
Fire	1
Intruder	2
.	.
.	.
Ringing Bell	N

Figure 5.14: Example content of a scenario priority list component. The lower the number the higher the priority of that scenario.

In this example when there are two scenarios selected, e.g. fire and intruder scenario, the one with the highest priority, e.g. fire scenario, will be executed first. This information is stored in an XML file providing the option to be modified easily without modifying the code of the main program. Anyone who wants to modify the priority list can open it with a simple text editor program and change the settings that one needs.

➢ Scenario selector

The incoming objects will be inserted in the scenario knowledge base system. Based on the results of the scenario knowledge base system the scenario selector component will select the most appropriate scenario. There is a threshold defined in this component. Any scenarios that the scenario knowledge base system selects have to be more than the threshold value. When multiple scenarios are selected the selector will choose the scenario with the highest priority based on the values in scenario priority list component. Figure 5.15 shows an example content of the scenario selector component.



Scenario Score Board

Scenario	*Chance*
1. Fire	75
2. Intruder	100
3. RingingBell	50

Figure 5.15: Example content of the scenario selector component. Fire scenario is at the first place because of its higher priority in the priority list, although its chance is lower than the intruder scenario.

➢ Short term memory and long term memory

The most advanced specifies on this world, the human beings have two kinds of memory components, the short term memory and long term memory. Imitating the memory concept of the human beings the AIBO will also use two memory components. The short term memory component contains the object and event that AIBO is dealing with at the moment. Figure 5.16 shows the two memory components and its content and their relationship.

First, when an object is recognized, it will be compared with the object in the short term memory. When the new object and its related event has a lower priority than the current object in the short term memory, the new object will be placed in the long term memory. Otherwise the new object will be placed in the short term memory and the old object which was in short term memory, will be placed in the long term memory. Duplication in the long term memory is not allowed. Therefore new objects which have exactly the same attributes as the objects in the long term memory, the old one will be removed. When AIBO encounters a lower priority object on its way to the goal point and also encounters it at the way back, the oldest object will be removed from the long term memory and a new one will be placed in the long term memory.

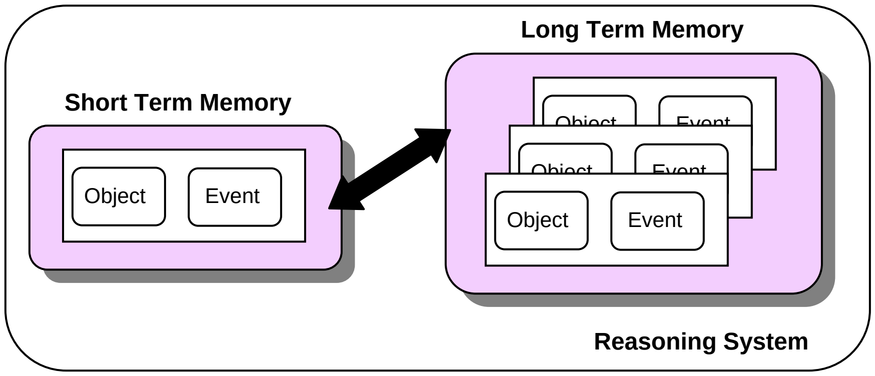


Figure 5.16: The content and structure of the short term memory and long term memory components. New incoming scenarios that have a lower priority than the scenario in the short term memory, will be placed in the long term memory.

The author has given this last-in-first-out approach for the long term memory above the approach based on the priority of the objects. The explanation for this choice is that when the current event consumes a lot of time the next event with the highest priority will probably not exist any more. Therefore it is no use to execute that event any more. Except the time issue there is also another reason to choose for the last-in-first-out approach, the place issue. When AIBO has just finished its last mission, the event of the last object is also the closest object to its current point. There is one exception for this rule: when AIBO has not encountered an object for a long time and AIBO is walking in a circle, then the closest one can be the oldest detected object. Figure 5.17 shows this situation. In general the last-in-first-out approach will be more efficient.

The best approach for the long term memory issue is a combination of the two approaches. We can set some threshold to determine which of the two approaches will be used. This can be based on the nearest place to satisfy a certain long term event or based on the priority level which has arrived within the last 5 minutes.

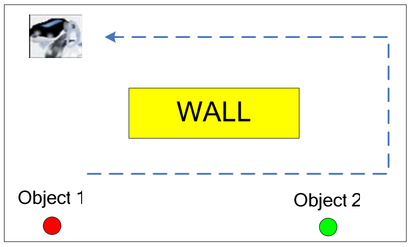


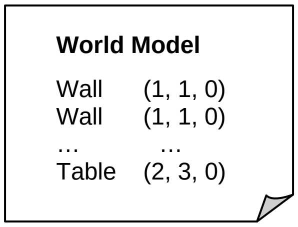
Figure 5.17: A situation that the used LIFO approach for executing the next scenario will not work very well. After making a round the closest object is not the last detected object, object 2, any more, but object 1. Therefore the next scenario should actually be object 1.

➢ Map

The map component deals with the world environment of the AIBO watchdog. The content of the Map component will be read from an XML file during the startup of the AIBO. This world model XML file contains the static objects of its environment. During patrolling of the AIBO the new incoming objects given by the manager component, will be inserted in the map component to keep the map up to date. Missing objects in the world environment can be noticed this way. If the manager component returns another object or no objects while the map is expecting a certain objects, this way changes in the world environment can be noticed.

➢ World Model - XML

In the world model objects are defined in the world environment of the AIBO watchdog. The XML file will be read when AIBO starts up. As a result AIBO will know what its environment looks like and where those objects are located. Figure 5.18 shows an example content of the XML file of the world model.



World Model

Wall	(1, 1, 0)
Wall	(1, 1, 0)
…	…
Table	(2, 3, 0)

Figure 5.18: Example content of the XML file of the world model. The numbers correspond to the x, y and z coordinates in the world environment respectively.

The structure of this file is fully dependent on the approach that is used to represent the world. This example shows the way point based approach of the world model.

➢ Event Handler

The event handler receives the changes of the current scenario or a new scenario will be introduced. Based on this information the event handler has to find an appropriate reaction. There is an action knowledge base system which will provide the necessary help for the event handler component to find the appropriate reactions. The event handler will also calculate the best path that the AIBO should take to fulfill its mission. The goal point will be provided by the action knowledge base system and the path will be calculated based on the information in the map component. Some certain reactions require some objects that can help the AIBO to fulfill its mission. For example the reaction is escaping this room; AIBO has to find in its passive memory for an object, e.g. a door, which can be used for escaping this room. This information can be provided by the manager component.

- **Action knowledge base system**

The action knowledge base system contains the actions that need to be carried out when a new object has been detected. This system can be created by an expert system or XML file. This system is just a look up table to find the appropriate actions. Figure 5.19 shows an example of a rule in the action knowledge base system based on an expert system.

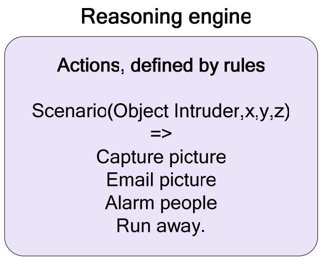


Figure 5.19: Example of a rule in the action knowledge base system based on an expert system. This rule will fire when an intruder object is detected. If this happens, the AIBO need to capture a picture, email it, alarm the surrounding people and run away to protect itself.

When the object intruder has been detected in an intruder scenario, the actions on the list have to be carried out. These actions will be put on the action stack component. It is chosen for the expert system approach, because of its speed of pattern matching.

5.3.3. Execution

The resulted actions from the reasoning process will be executed by the execution components. This component is the interaction component with its environment.

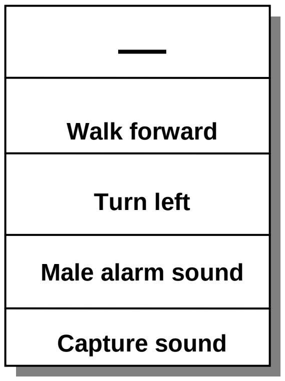


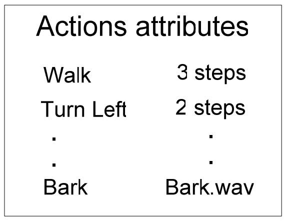
Figure 5.20: Example of the content in an action stack component. The actions at the bottom will be executed first. It is based on the first in first out concept.

Action stack

The action stack contains the resulted actions from the reasoning process. These action needs to be executed in the first-in-first-out order. Figure 5.20 shows the possible content of the action stack. The actions at the bottom will be executed first and new actions will be placed on the top.

Actions Attributes – XML

This XML file contains the attributes of the actions that need to be carried out. As a result actions can be slightly modified by these attributes. Figure 5.21 illustrates the content of this XML file.



Actions attributes

Walk	3 steps
Turn Left	2 steps
.	.
.	.
Bark	Bark.wav

Figure 5.21: Example content of the Actions Attributes XML file. This file describes the attributes of the movements. In this example a walk movement consists of 3 steps and the barking sound is located at Bark.wav.

By changing the values of the action attributes, AIBO will walk farther or turn less. The bark sound can also be modified by a better sound sample. This XML file approach provides an easier modification process of the actions attributes. This is especially useful when AIBO needs to operate in a new environment which will give difficulties with the current attribute values.

Executor

The executor component deals with the execution of the actions. It communicates with the internal operating system of the AIBO and actions will be translated to the syntax of the internal operating system of the AIBO. First it fetches the action that needs to be carried out from the action stack and thereafter the matching attributes will be found in the actions attributes xml component. This information will be translated to the language that the software of AIBO can understand. The last step will be transferring this information to the AIBO. Based on this information AIBO will carry out the physical actions.

5.4. Home Environment

In this section the ideal home environment and the test environment for the prototype will be discussed.

5.4.1. Ideal Home Environment

The ideal environment which can be derived from our environment specification is shown in Figure 5.22. It shows a modal home environment where the AIBO watchdog can patrol in future, but for testing the prototype a simpler home environment will be chosen.

Figure 5.23 shows a simple home environment where the AIBO watchdog can operate. It consists of a living room with the standard objects that can be found in every average family. Further more the home environment consists of a simple bedroom with its necessities, an average kitchen and a bathroom which provides simple sanitary facilities. The objects that are found in the rooms of the home environment are the most important objects that characterize that specific room.

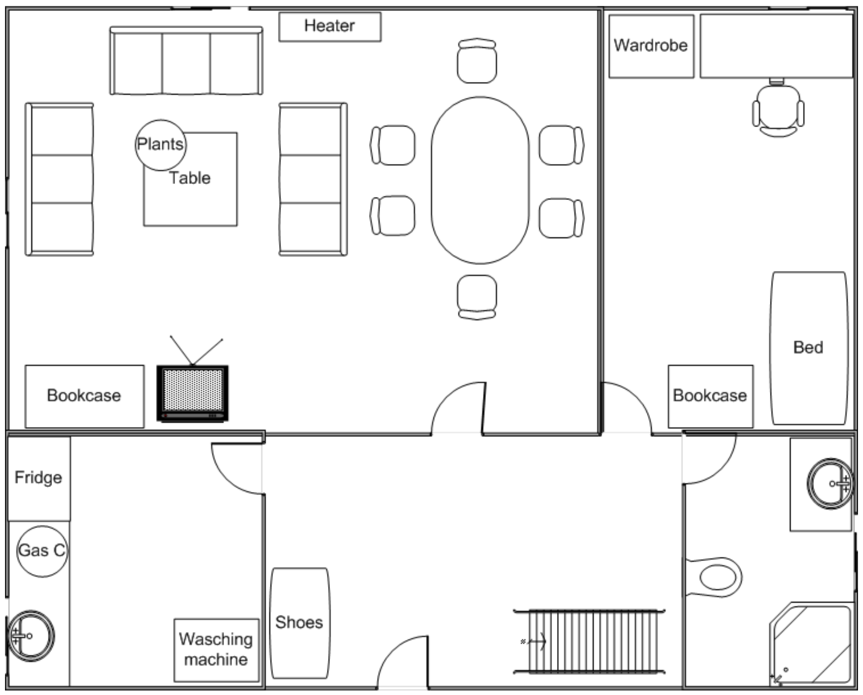


Figure 5.22: An example representation of the ideal home environment where the AIBO watchdog has to take care of.

5.4.2. AIBO Navigation

To test the developed architecture of the AIBO watchdog a simpler home environment will be used. Figure 5.23 shows the map of the simpler home environment. There are objects from the environment world in the test environment. For the navigation of AIBO the area are covered by waypoints. The start point and endpoint of AIBO walking are the small circle dots on the map. These small dots will be given beforehand to the AIBO.

The navigation goals are represented as the small squares with a character above it. The next destination node is default the next character in alphabetic order. When special situations arise, the goal node will be adapted. The path between the two nodes will freely be chosen by the route planner, but it will take the node with the shortest distance to the destination node by default. Except when there are objects on its way, it will take another route to avoid collision.

Since this project is not dealing with the pre processing of the sensors input, the input of the sensors will be simulated and stored in the nodes of the waypoints. An external event generator component which is part of the GUI module will simulate the special events and possible scenario's which will change in time. In summary all input features of the sensors will be stored in the nodes and these features can be modified by the user using the GUI module. This is designed to be able to create the special objects and its corresponding scenario.

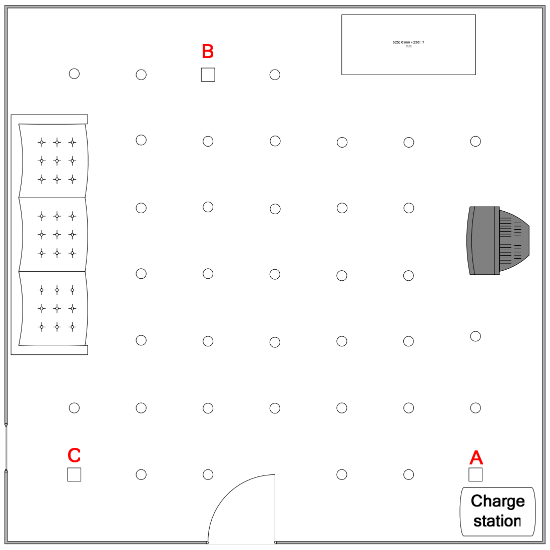


Figure 5.23: A picture representation of the test environment for the AIBO watchdog project.

6 Software Design

"There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make so complicated that there are no obvious deficiencies."
C.A.R. Hoare

This chapter presents UML models of the AIBO watchdog. First a brief introduction about UML will be given and thereafter the UML models about the AIBO watchdog will be discussed. The Unified Modeling Language (UML) is a modeling language for specifying, visualizing, constructing, and documenting the artifacts of a system intensive process. It was originally conceived by Rational Software Corporation and three of the most prominent methodologists in the information systems and technology industry, Grady Booch, James Rumbaugh, and Ivar Jacobson (the Three Amigos). The language has gained significant industry support from various organizations via the UML Partners Consortium and has been submitted to and approved by the Object Management Group (OMG) as a standard (November 17, 1997) [24].

Diagrams depict knowledge in a communicable form. The UML provides the following diagrams, organized around architectural views, regarding models of problems and solutions:

- The User Model View
  - Use case diagrams depict the functionality of a system.
- The Structural Model View
  - Class diagrams depict the static structure of a system.
  - Object diagrams depict the static structure of a system at a particular time.
- The Behavioral Model View
  - Sequence diagrams depict the specification of behavior.
  - Collaboration diagrams depict the realization of behavior.
  - State diagrams depict the status conditions and responses of participants involved in behavior.

- Activity diagrams depict the activities of participants involved in behavior.

➢ The Implementation Model View

- Component diagrams depict the organization of solution components.

➢ The Environment Model View

- Deployment diagrams depict the configuration of environment elements and the mapping of solution components onto them.

Fundamentally, diagrams depict knowledge (syntax). Because the foundation of the UML constitutes the necessary and sufficient engineering practices for problem solving, processes that utilize the UML are assured of resting upon a foundation that provides the potential for success [24].

6.1. Use-Case Diagram

The Use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The Use case diagram shows which actors interact with each use case. In other words it captures the functional aspect of a system [25]. Figure 6.1 shows the global use-case diagram of the AIBO watchdog.

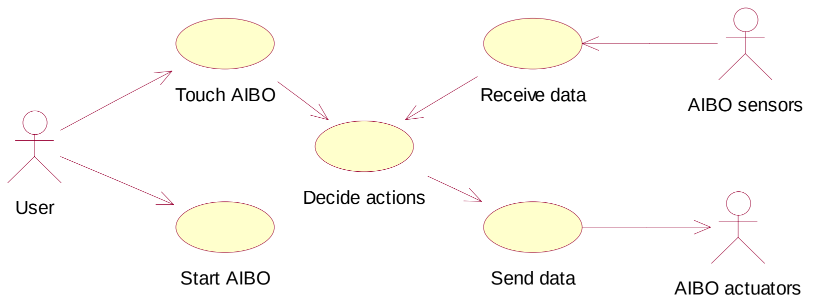


Figure 6.1: Use-case diagram of AIBO watchdog.

<u>Actors:</u>

- User –The user of the AIBO who can start the AIBO.
- AIBO sensors – These are the sensors of the AIBO watchdog which can perceive its environment.
- AIBO actuators – The output actuators of AIBO watchdog that interacts with its environment.

<u>Use-cases:</u>

- Start AIBO – The user can start the AIBO by switching the power-button on. The boot program on the memory stick will be loaded.
- Touch AIBO – By touching the sensors of the AIBO watchdog certain actions will be executed.
- Receive data – During perceiving the environment the sensors will get the raw input data of the objects in the environment.

- Send data – When the reactions have been determined the commands will be sent to the AIBO actuators.
- Decide actions – Based on the input from the sensors reactions will be determined.

The table below shows the decide actions use-case:

Table 6.1: Use-case diagram, decide actions.

Use-case name	Decide actions
Participating actors	Initiated by user or AIBO sensors.
Entry condition	1. Sensors input data are received initiated by environment or user.
Flow of events	2. The noise and redundant data from the received sensors data need to be filtered out. 3. From the filtered data the features of the environment need to be processed. 4. Objects will be recognized from these features. 5. These recognized objects will be sent to the reasoning system to determine the current situation. 6. The reasoning system will produce a list of actions that are needed to be executed. 7. This list of actions will be converted into command data that the AIBO actuators can interpret.
Exit condition	8. The reactions will be executed by the AIBO actuators.

6.2. Subsystem Decomposition Diagram

The total system of the AIBO watchdog can be decomposed into several subsystems. These subsystems can be implemented and replaced separately, but they are all necessary to let the AIBO watchdog work appropriately. Figure 6.2 shows the subsystems of the AIBO watchdog.

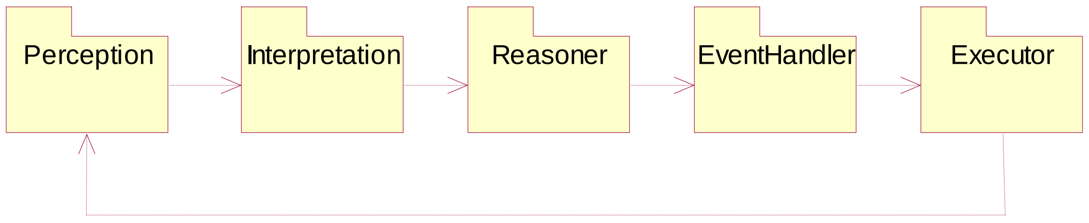


Figure 6.2: Subsystems of the AIBO watchdog. Each of them can be implemented separately, but they are all needed to make AIBO watchdog working. A missing subsystem will break the chain.

If one of the subsystems is missing, the chain will be broken. The AIBO watchdog will not function properly. Therefore it is necessary to create a simplified version of each component that we do not want to focus. Simulation of a subsystem can be an option.

Perception

The perception subsystem deals with the sensors input data, e.g. data from camera or microphone. The data from a camera can be a picture or video. From the picture it is required to extract the useful features. Color and shape are the most useful ones. From the video features of moving objects can be extracted.

Interpretation

When the features are extracted from the raw sensor's input, the system needs to recognize the objects. Features alone do not provide any meaning. When the object has been recognized a useful meaning can be assigned to that particular object.

An object recognition program can fulfill the perception and interpretation subsystem at the same time.

Reasoner

The reasoner subsystem deals with the recognized object by the interpretation subsystem. Based on this object a certain hypothesis has to be made. When there are sufficient evidences, it will become a scenario. If there are more than one scenario, prioritizing these scenarios will be also a task for the reasoner subsystem. The most important one will be handled first and the other ones will follow thereafter. Scenarios will be solved by the eventhandler subsystem.

EventHandler

The eventhandler subsystem receives the most important scenario and solutions needs to be found. The solutions are the reactions that AIBO watchdog needs to execute. Physical movements of the AIBO, e.g. turning, will be calculated by the navigation algorithm. This is required to understand commands like, go to the owner. Therefore the navigation algorithm for calculating the path to a certain goal point is also included in this subsystem. The resulted reactions will be put on the stack.

Executor

The executor subsystem takes care of the execution of actions that are retrieved from the stack which are put by the eventhandler subsystem. Actions will be retrieved from the stack one by one. The next action will be executed when the previous action has already been completed, but there is an exception on this rule. Two or more actions can be executed in parallel, if they are independent of each other, e.g. creating sounds and walking at the same time. The actions must be able to be modified by changing its parameters. On a different floor AIBO walks differently, so sometimes it needs more steps to turn a full round than normally.

6.3. Class Diagram

A Class diagram gives an overview of a system by showing its classes, its attributes and the relationships among them. Class diagrams are static. They display what interacts, but not what happens when they do interact [26].

A class in the software system is represented by a box with the name of the class written inside it. A compartment below the class name can show the class's attributes (i.e. its properties). Each attribute is shown with at least its name, and optionally with its type, initial value, and other properties. The class's operations (i.e. its methods) can appear in another compartment. Each operation is shown with at least its name, and optionally also with its parameters and return type.

6.3.1. Class Diagram of Environment

The chosen approach for acquiring the content of the environment was importing them partially from an external file. All static objects will be read from a clip file which can be changed by using a normal text editor program. As a result the home environment can easily be changed by modifying the clip file instead of modifying the internal program code.

The special objects are not read from a file, but created by the user. We can create real fire on certain parts of the home environment and let the AIBO detect it or create a virtual fire object in the reasoning system. Since this project is focusing on the reasoning system and not the preprocessing of the sensors inputs. The second approach will be chosen.

The user manipulates the data in the reasoning engine directly and creates a virtual fire object in the reasoning knowledge database. When AIBO reaches a point that contains a fire object, AIBO will carry out the appropriate reactions. Figure 6.3 shows the class diagrams of the environment part.

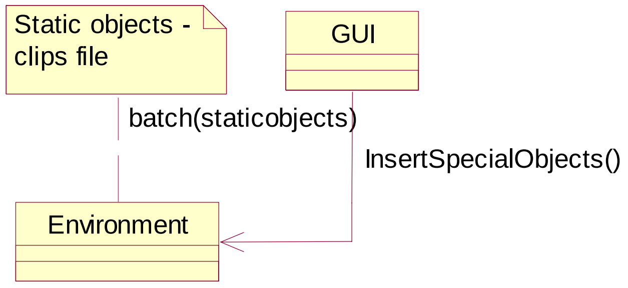


Figure 6.3: Class diagram environment.

The special objects will be inserted by the user itself to create certain events. The GUI will help the user to accomplish this task easily.

6.3.2. Features World

The features world will convert the objects in the environment into features which are sensible by the AIBO watchdog. Therefore only features which concern the image and sound sensor will be used. Normally this process will be carried out by the preprocessing unit of the sensors. Figure 6.4 shows the relationship between the environment class and the AIBO features class.

The features that will be used are color, shape, material, sound amplitude and sound frequency and wavelength. In the test scenario these features will be assigned to every object.

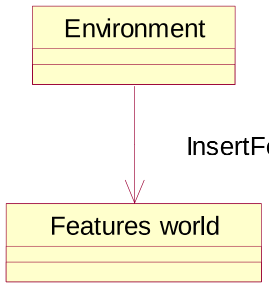


Figure 6.4: Relationship between environment and features world in class diagram.

6.3.3. Interpreted World

After having the features of a certain object the AIBO will try to recognize the perceived object. The recognized object is not necessary the same one as the one perceived in the environment. This interpretation error is the cause of ambiguity in the environment world and the few features that AIBO can sense. Figure 6.5 shows the relationship between the features world and the interpreted world.

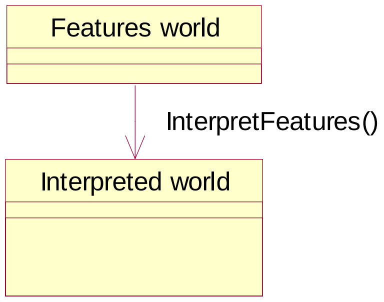


Figure 6.5: Relationship of interpreted world and features world in class diagram.

The interpreted world is in the ideal situation the same as the environment, but because of sensing errors, noise and limited classification capacity of the programs, the Interpreted world is a simpler and possibly not identical representation of the environment.

6.3.4. Scenario Reasoning System

The scenario reasoning system deals with the reasoning of the objects that the AIBO perceive at a certain location. Depending on the new incoming object a specific scenario will be concluded. If this scenario has a higher priority, this scenario will be solved first. Figure 6.6 shows the complete class diagram of the scenario reasoning system and its related classes.

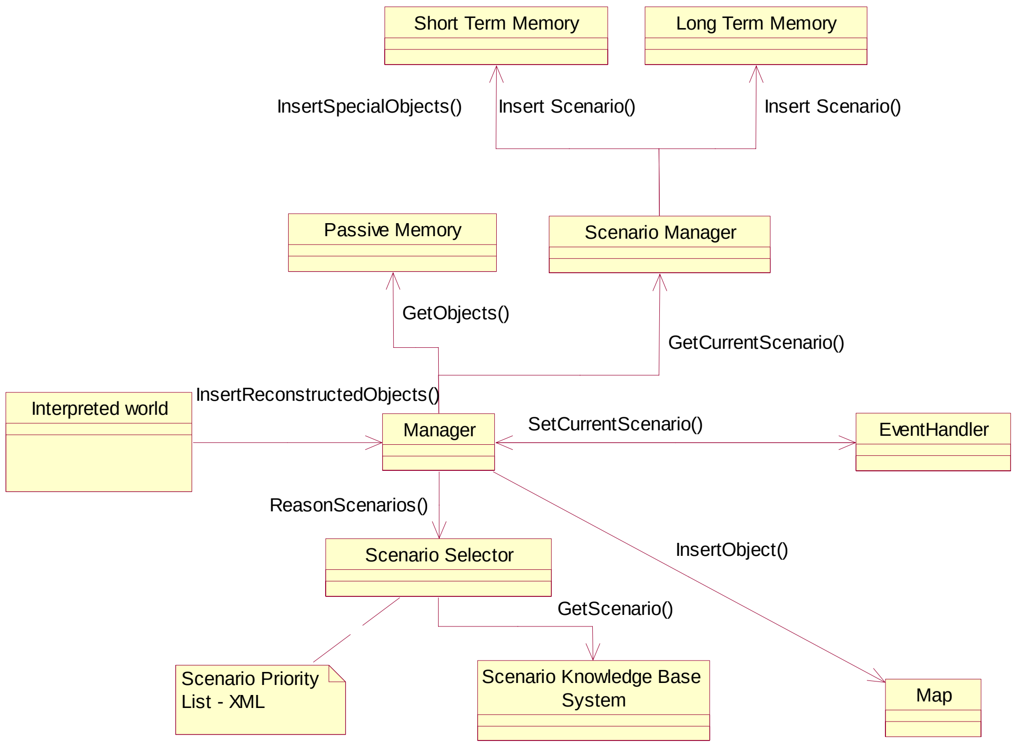


Figure 6.6: Class diagrams of the scenario reasoning system.

The manager receives the recognized objects from the interpreted world class. These objects will be stored and ordered in the passive memory class if these objects are useful. The incoming objects will be inserted in the map class and the world environment of the AIBO will be updated. The recognized object will also be passed to the scenario selector class and thereafter to the scenario knowledge base system. This system determines the scenarios based on the incoming objects. After matching these scenarios with the scenario priority list the most important scenario will be selected and passed to the manager class. Depending on the other unfinished scenarios that were detected in the past the current scenario will be put on the short term memory or long term memory. The scenario in the short term memory, which is the most important scenario at the moment, will be passed to the EventHandler class.

6.3.5. Action Reasoning System

After understanding the current situation actions are needed to solve the current situation. The kinds of required reactions are determined by the action reasoning system. Figure 6.7 shows the class diagram of the action reasoning system and its related classes. The class EventHandler receives the decided scenario from the manager class. This scenario will be passed to the Action Knowledge Base System class to determine the actions that are required to be executed. A matching in the Action Knowledge Base System will be executed and the results will be passed to the EventHandler. The resulted actions will be inserted in the class ActionStack. If a moving action was a part of the resulted actions, the moving direction of AIBO will be calculated by the Navigator class.

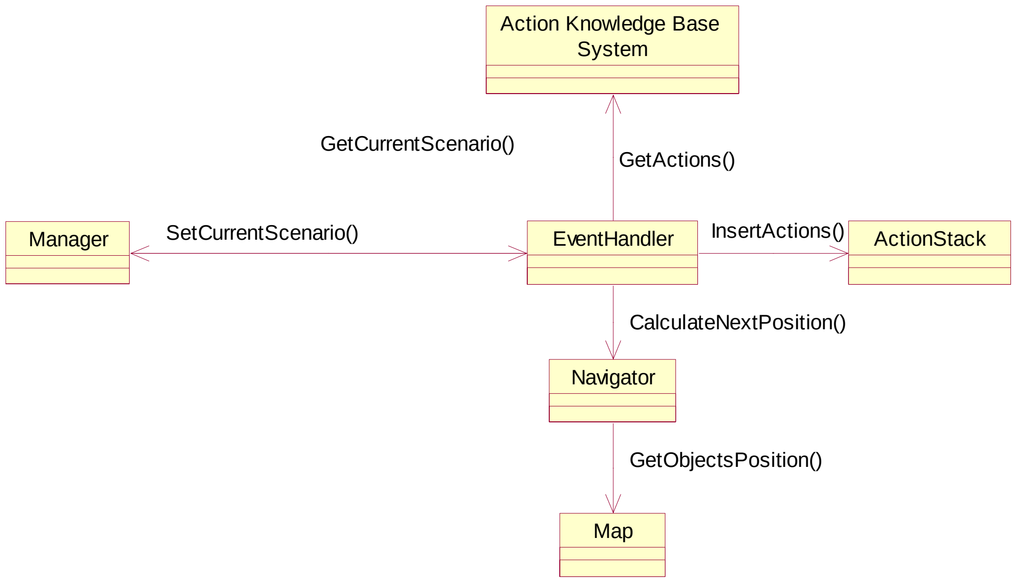


Figure 6.7: Class diagrams of the action reasoning system.

6.3.6. Actions Execution

When the actions are determined and put on the action stack. The next step, also the last step, will be the execution of these actions. Figure 6.8 shows class diagrams of the execution of actions.

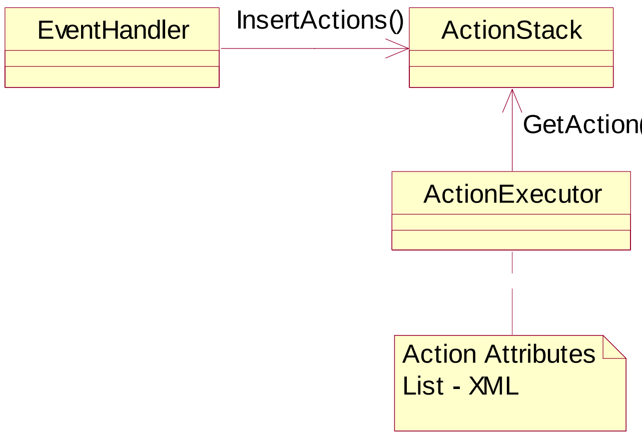


Figure 6.8: The class diagrams of the execution of actions. Class ActionExecutor fetches the actions from the ActionStack and they will be executed one by one. The action attributes list XML file contains the parameters of the actions.

The class ActionStack contains the actions that the class EventHandler has put on the stack. The class ActionExecutor fetches these actions from the ActionStack and these actions will be executed one by one. The actions parameters, such as the number of steps for a walk function or the directory that the picture needs to be saved will be obtained from the Action Attributes List XML file.

6.3.7. Total Class Diagram

Every time when there is something detected, this whole process will be carried out, from sensing to the actions. This process will never end; it is a continuous and never ending process. The main reasoning system is based on the reasoning engine and the manager. This reasoning engine is based on facts and rules, which are if, then, else rules. This reasoning engine is a very deterministic system which will not always give the best result. It is possible to use another reasoning engine based on the probabilistic model such as a Bayesian network. The reasoning engine class has to be replaced by the new model and the activity manager has to be adapted. The total class diagram is shown in Figure 6.9.

6.4. Sequence Diagram

The sequence diagram shows the behavior of every class during the reasoning process. Since the class diagrams are already presented the interaction process between the classes can easily be demonstrated.

Figure 6.10 illustrates the sequence diagram about the reasoning process between the reasoning classes.

This sequence diagram shows the reasoning process of a new incoming object. This new object triggers a new scenario. The resulted scenario has a higher priority than the scenario that was present in the short term memory. Therefore the new scenario will be placed on the short term memory and the scenario that was present in the short term memory will be placed on the long term memory. The new scenario will also be passed to the EventHandler, so the reactions on the new scenario can be determined. This sequence diagram shows only a part of the total sequence diagram. Only the reasoning process in the scenario reasoning system has been shown.

Figure 6.11 shows the next part of the sequence diagram. This sequence diagram shows the interaction process after determining the scenario.

This sequence diagram continues the explanation of the previous sequence diagram. The actions will be determined for the current scenario. In this scenario AIBO needs to walk to a certain place, therefore a walk path needs to be calculated. These actions will be put on the ActionStack. From this stack the ActionExecutor will fetch the reactions that are required to be executed. From the ActionAttributesList the action attributes will be obtained and the appropriate action with its corresponding attributes can be executed by the ActionExecutor. This sequence diagram shows only one execution loop by the ActionExecutor, but the idea is that it has to execute all actions on the stack.

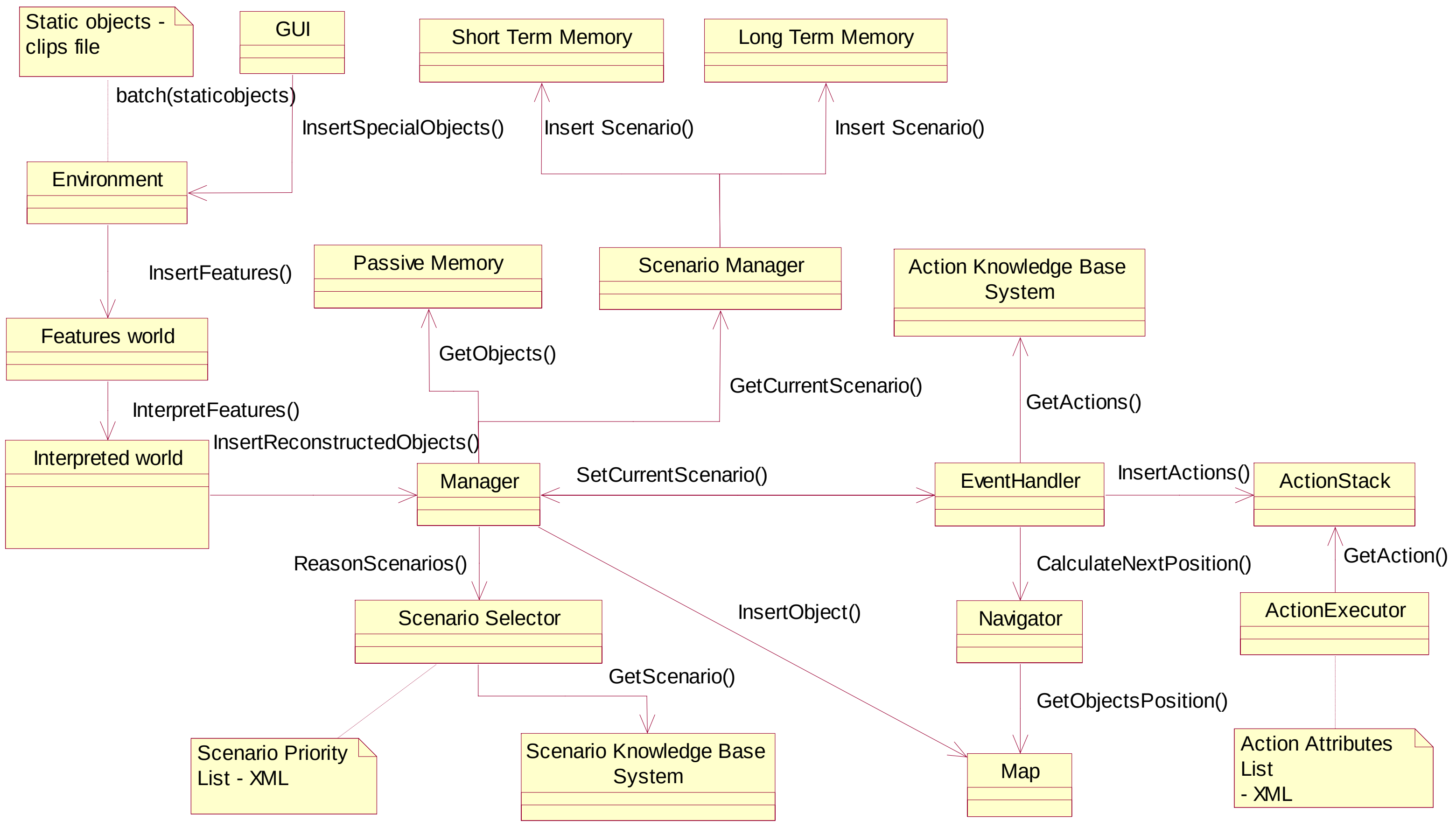


Figure 6.9: Total class diagram, design of AIBO watchdog.

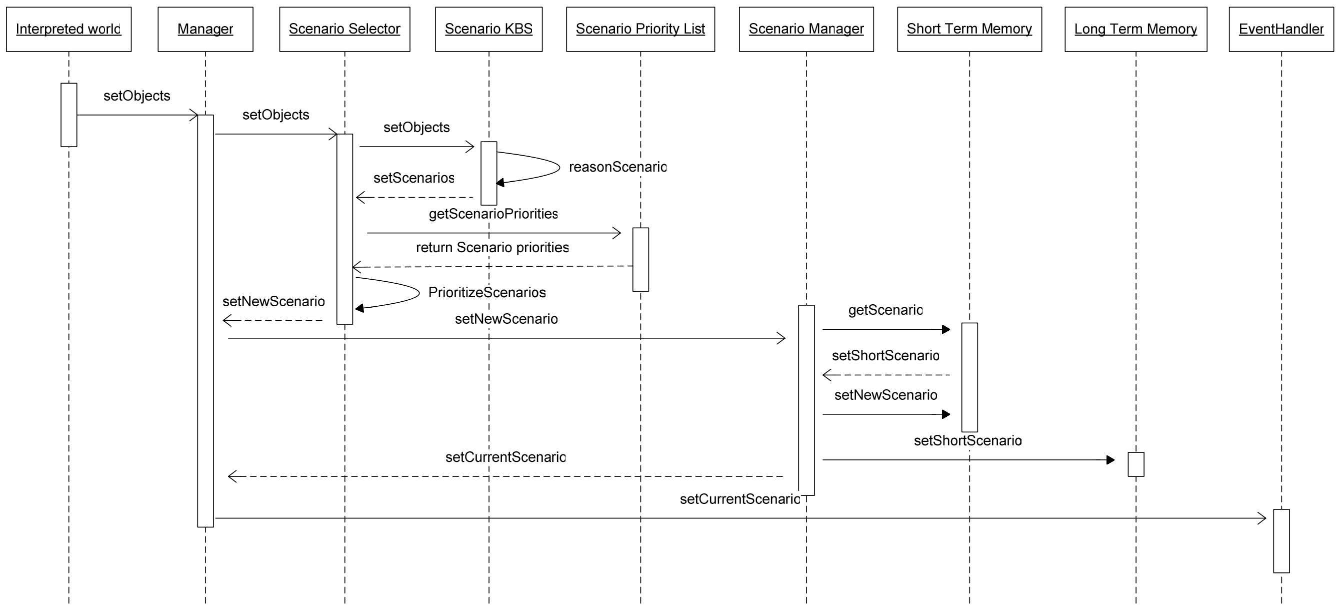


Figure 6.10: A part of the sequence diagram of the reasoning classes. This sequence diagram shows the reasoning process of a new incoming object. The resulted scenario has a higher priority than the scenario that was present in the short term memory.

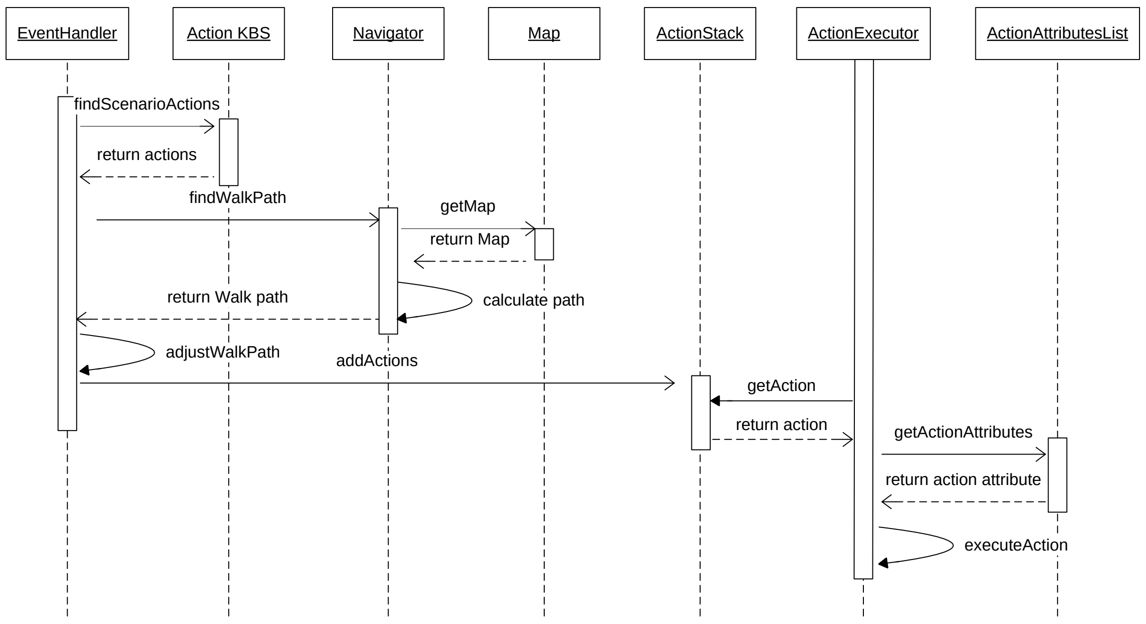


Figure 6.11: The sequence diagram of the Action reasoning system and the actions execution. Actions are determined and the walk path for the AIBO will also be calculated. Actions are put on the stack and executed according to the action attributes.

6.5. Flowchart

The flowchart diagram provides the inside view of the working of the reasoning process. By following the flowchart reasoning decisions can be understood. Figure 6.12 shows the simplified flowchart of the reasoning process of the AIBO watchdog.

During the initialization process the AIBO will be started and the data will be loaded from the program. Its first task is to explore the environment of the current node. When something is detected, the appropriate actions will be executed. Otherwise the route planner system has to calculate the next node that AIBO has to move to. After deciding the next node, AIBO will turn itself to face that direction. Before the actual movement can be executed, AIBO has to use its distance sensor to get the latest map information. This can be the case when suddenly some objects appeared to block its way. Then another route will be calculated. If the way is free to move, AIBO will move to that position. When AIBO has arrived the goal node, the route planner has to be updated. Otherwise AIBO can carry on moving itself to the goal position.

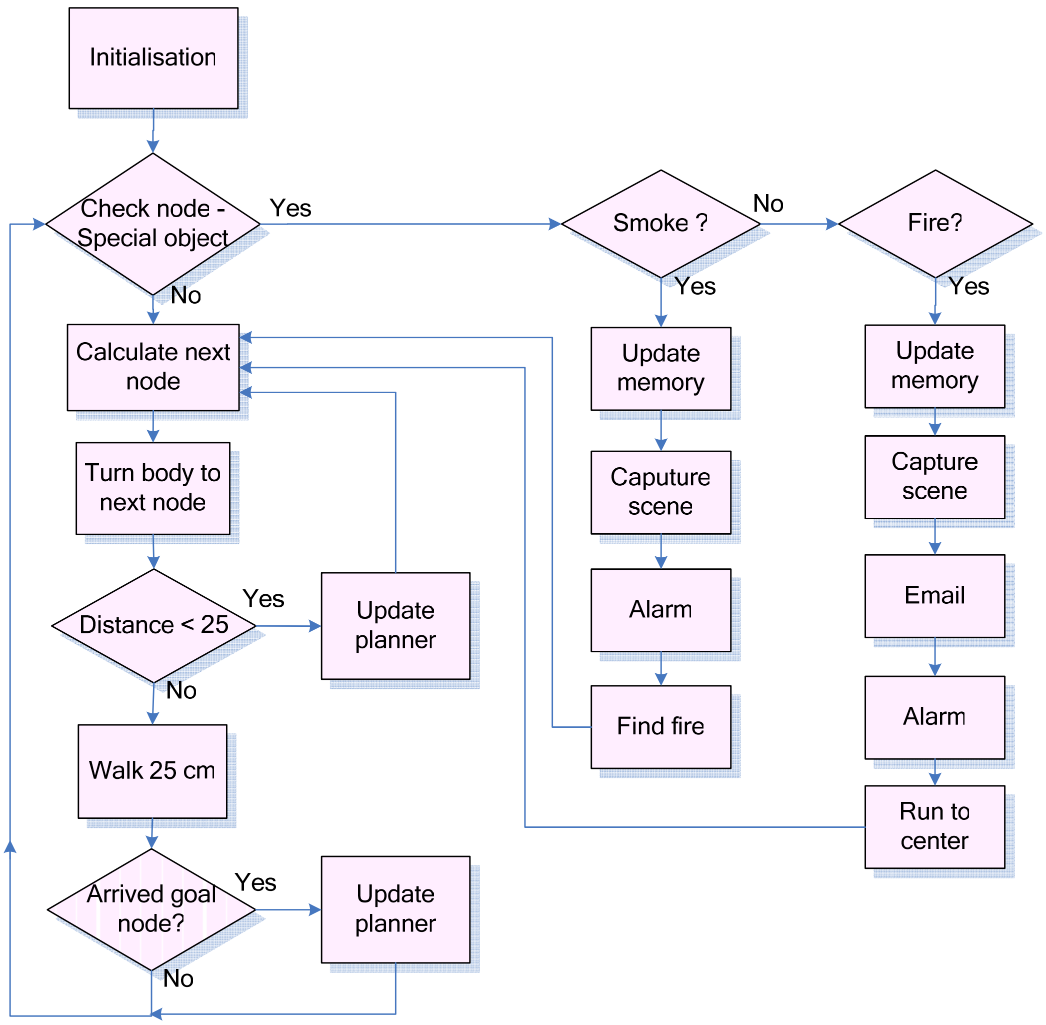


Figure 6.12. Simplified flowchart of the reasoning process.

7 Implementation

"All that we are is a result of what we have thought."
Abraham Lincoln

This chapter describes the actual implementation of the AIBO watchdog. First an overview of the used software and tools are given and afterwards the implementation of the AIBO watchdog is discussed. As indicated in the previous chapters every component in the architecture design will be implemented and sometimes a simplified method will be chosen. At the end of this chapter a manual will be given to use and modify the program.

7.1. Used Software & Tools

- **Java & Eclipse**

For the implementation of the overall architecture it was chosen for Java. Java was chosen above C++, because the predecessor of the AIBO watchdog project has chosen for Java. It was better to start from where the predecessor has stopped and adapt the code when necessary. By reusing a part of the code it saved a lot of time.

The development environment was chosen for Eclipse. Eclipse is a reasonably complete development environment that can speed up the development process. It provides a lot of functionalities for Java programming. The program has a code editor that offers code assistance, e.g. the possible methods of an instance will be shown after typing the name of the instance. Beside this, it also provides multiple overviews of the code, which makes it easy to jump to specific parts of the code. Because of its broad functionalities and its speed compared to other development environment, the author has chosen for Eclipse. The used version of Eclipse for the development of the program was Eclipse 3.1.2. The used Java version was based on Java runtime environment (JRE) 1.4.2.

- **User Interface & Netbeans**

For presenting and manipulating the simulation of the AIBO watchdog environment it was chosen for the Java development environment Netbeans. By its drag and drop functionality it was relatively simple to place the corresponding buttons and fields on the user interface. Its automatic aligning functionality provides a good tool to create a decent user interface. Netbeans provides only a good tool to create user interfaces. It is very weak at the other usability area of programming, e.g. it is not possible to adapt the user interface by programming codes. Other weaknesses of Netbeans are its slowness, its huge requiring of processor intensity and its huge consuming of the working memory. But for designing the code for user interfaces Netbeans is a better choice. Eclipse does not provide it at all. Netbeans 5.0 has been used for creating the user interface, which was the newest version at the moment of development.

- **Expert System & Jess**

In order to develop a reasoning system rapidly it has been chosen for expert system above all other reasoning systems in the Artificial Intelligence world. It is because of its ease of use and easy to understand. If we have already chosen for Java, Jess, Java Expert System Shell, is the best option. Java can be used within Jess or vice versa. Jess itself is written in Java, therefore it provides us the best integration with our java framework. The Jess files can easily be modified with a simple text editor program, such as notepad. For the development of the program Jess version 7.0 has been used. More information about Jess can be found at appendix A.

- **Database & XML**

XML, eXtensible Markup Language, is a very popular format to store information for websites, because of its flexibility. XML files can easily be modified by a simple text editor program, such as notepad, which can be found standard on every windows pc. XML is actually a small database where websites can put its temporary data, e.g. the settings of the layout of a website. Therefore we have also used XML to store the settings of the movements of the AIBO watchdog and other settings which need flexibility. The flexibility that XML provided, allows the user to modify the settings of the program and create their own AIBO watchdog.

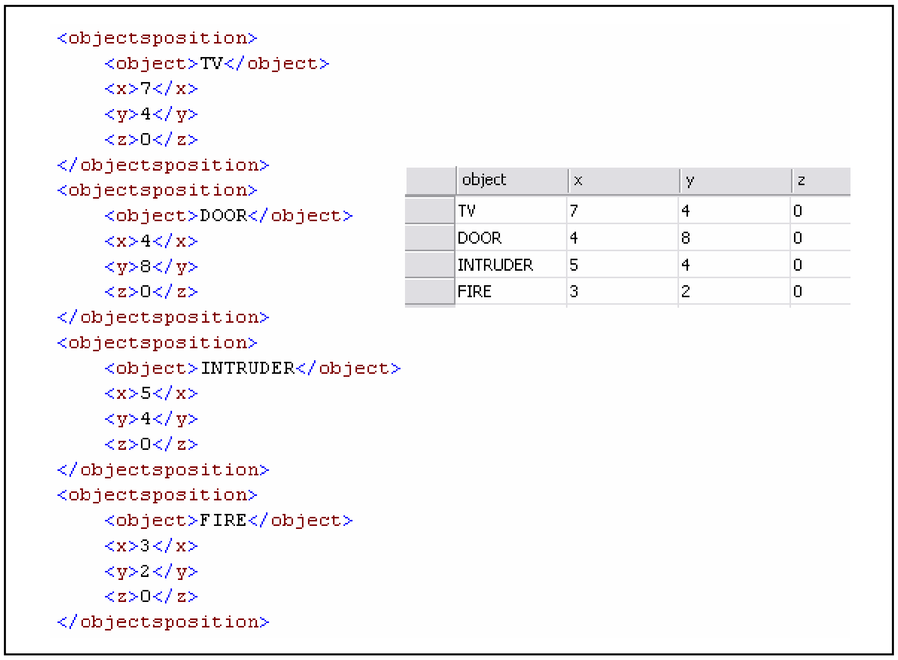
- **XML & Microsoft Visual Studio .Net 2003**

For the best representation of the XML files it is chosen for Microsoft Visual Studio .Net 2003. XML codes will be converted to tables which is visually better readable than a large text file. Modifications need fewer actions than a text file, because the tags will be standard created or removed during modifications. This program helps us to save a lot of time. Figure 7.1 shows the XML representation in Microsoft Visual Studio .Net 2003.

- **Hardware requirements**

In this part of the section explanations will be given about the hardware requirements that are needed to run the software. The hardware specifications of the AIBO are fixed; therefore changes on the AIBO can not be made. For the project the AIBO-ERS7 has been used, it is necessary to use the same AIBO for running the program. Otherwise it can cause malfunctions or compatibility problems since the old versions of AIBO offers less sensors input and another software environment.

The used client and server architecture requires an external computer to perform the processor and memory intensive programs. This computer needs to have a wireless network card which is compatible with 802.11b. A peer-to-peer connection with the AIBO is made, so a wireless access point is not needed. Furthermore a memory stick reader is necessary to transfer the program to the memory stick. The speed that the AIBO will react depends on its wireless communication speed with the external computer and the speed of the external computer that executes the program. The software was developed using an Intel Centrino laptop which provides 768MB working memory and 1.86 Ghz processing power. During execution everything was running very smooth. So it is recommended that the external PC has at least 512MB working memory and at least a Pentium 4 processor for processing.



```
<objectsposition>
    <object>TV</object>
    <x>7</x>
    <y>4</y>
    <z>0</z>
</objectsposition>
<objectsposition>
    <object>DOOR</object>
    <x>4</x>
    <y>8</y>
    <z>0</z>
</objectsposition>
<objectsposition>
    <object>INTRUDER</object>
    <x>5</x>
    <y>4</y>
    <z>0</z>
</objectsposition>
<objectsposition>
    <object>FIRE</object>
    <x>3</x>
    <y>2</y>
    <z>0</z>
</objectsposition>
```

object	x	y	z
TV	7	4	0
DOOR	4	8	0
INTRUDER	5	4	0
FIRE	3	2	0

Figure 7.1: Advantage illustration of XML editing with Microsoft Visual Studio. At the left side the code of the XML are presented. At the right side the spreadsheet-like table is the Microsoft Visual Studio .Net 2003 representation of the XML code.

7.2. Implemented Architecture

This section discusses the implementation of the AIBO watchdog. First an overview of the implemented components of the architecture will be provided. Thereafter an overview of the implemented system will be given and afterwards a more detailed description of each component will be presented.

7.2.1. Overview of the Implemented Architecture

The architecture that is described in the design chapter has been used for the implementation of AIBO watchdog. Figure 7.2 shows the implemented components of the architecture. The components that have a gray color have been implemented. The white components on the other hand have not been implemented.

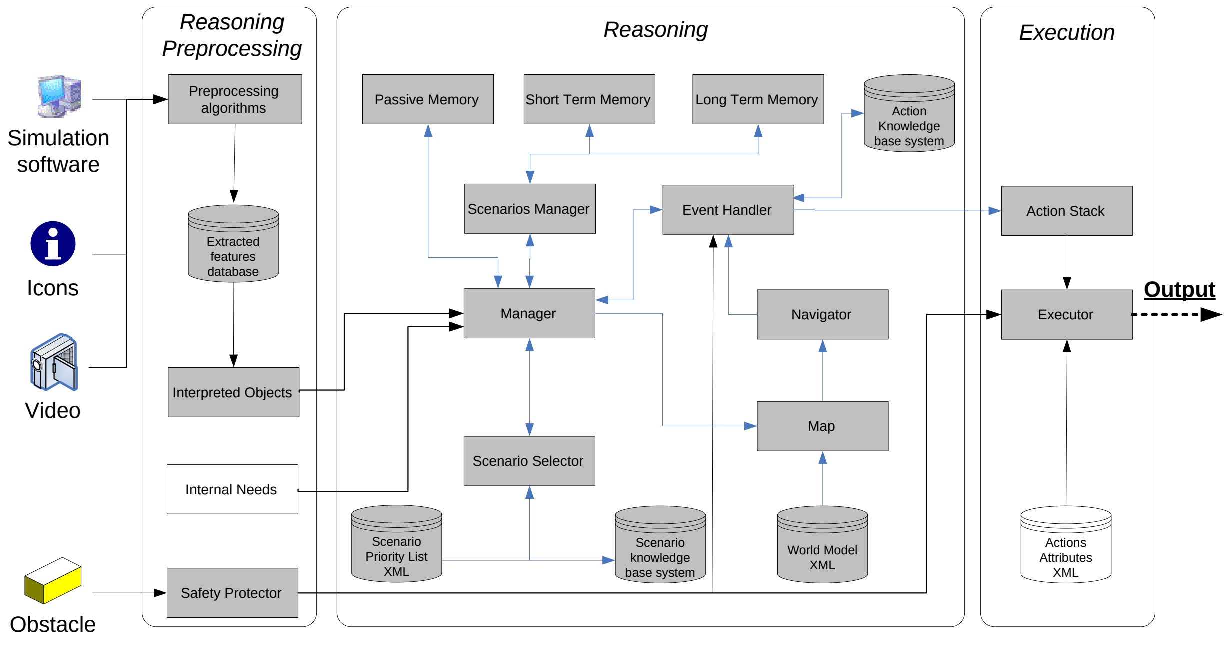


Figure 7.2: Implemented components of the architecture. The gray components have been implemented.

7.2.2. Overview of the System

The implemented framework is based on the design architecture of the AIBO watchdog. Figure 7.3 illustrates the implemented framework.

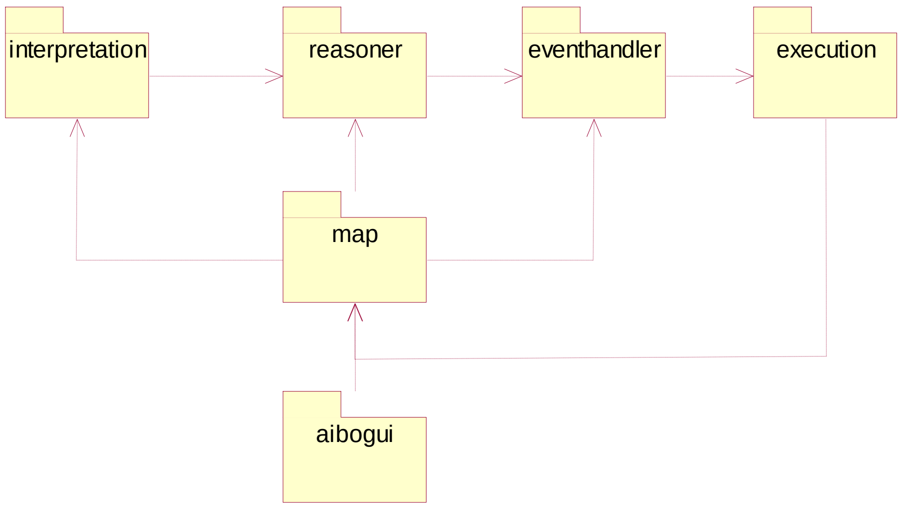


Figure 7.3: Overview of the packages in the implemented framework.

If we compare this implemented framework with the design architecture of the AIBO watchdog, we can discover a lot of similarities. The packages in the implemented framework will be explained in detail in the following subsections.

➢ AIBO GUI

The AIBO GUI is required to test our AIBO watchdog. Ultimately our AIBO watchdog is an autonomous agent without user interface. The AIBO GUI converts the AIBO mind into a visual representation that is understandable for human beings. The package AIBO GUI consists of all available user interfaces to interact with the AIBO watchdog. In total there are three GUI's: Main GUI, Legend GUI and Object GUI. Figure 7.4 shows the Main GUI.

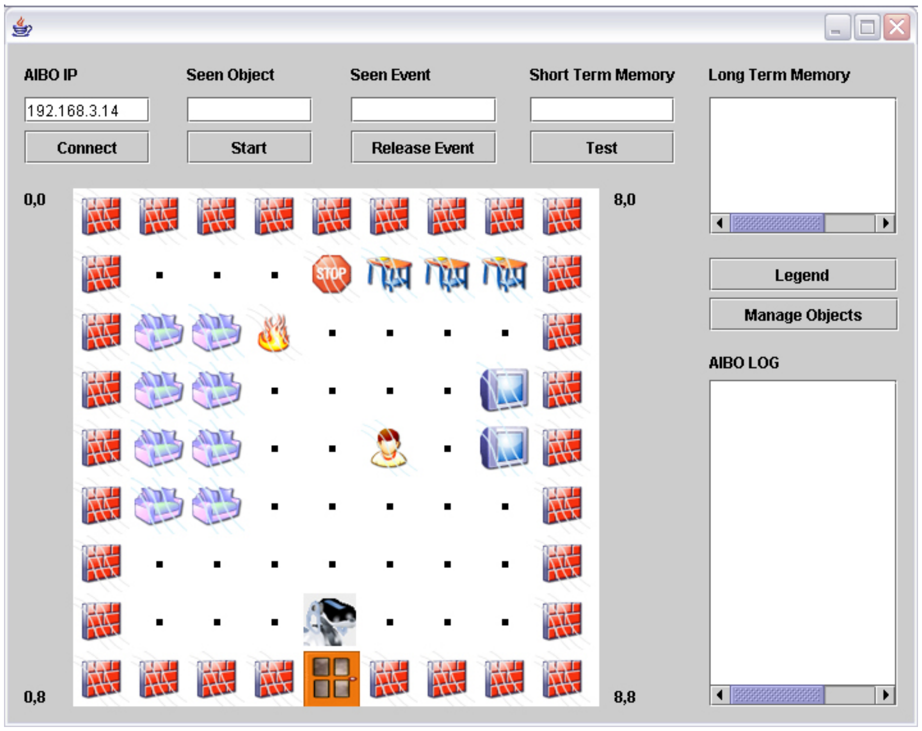


Figure 7.4: Interface of the Main GUI. The picture at the left side is the representation of the simulation of the environment. The other fields will display a certain value so the user can track the patrolling of the AIBO watchdog and know what has happened.

At the top left side one can fill in the IP of the AIBO. This way a peer-to-peer connection will be realized. The objects that AIBO encounters during patrolling will be shown in the seen object field. Every seen object can trigger an event. The corresponding event will be displayed in the seen event field. Every event that is triggered will be saved in the short term or long term memory. This is fully dependent on the number of current events and the priority of the new one. The AIBO log area at the right side will display the events at every point that AIBO has passed by. The picture in the middle of the Main GUI shows the map of the corresponding environment where AIBO is patrolling. The environment is divided in waypoints and at each waypoint there can be an icon present. The meaning of the icons can be found in the Legend GUI. Figure 7.5 shows the interface of the Legend GUI.

The Legend GUI is nothing more than a legend to find the meaning of the used icons. It functions as a lookup table for the user.

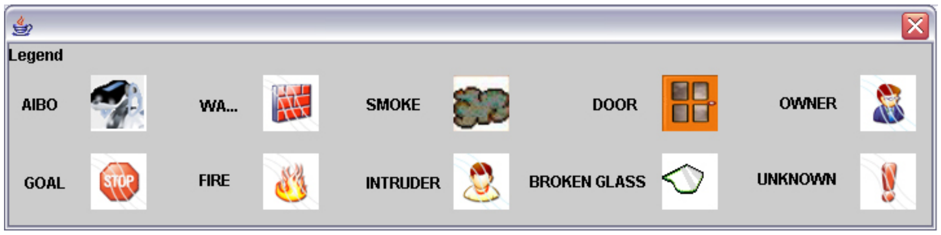


Figure 7.5: Interface of the Legend GUI. The Legend GUI shows the meaning of every icons that is used in the environment representation of the MainGUI.

The third GUI, Object GUI, provides the ability to change the simulation environment of the AIBO watchdog. Figure 7.6 shows the Object GUI.

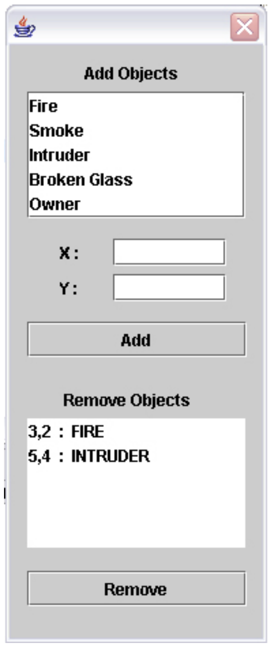


Figure 7.6: Interface of the Object GUI. This GUI can modify the objects in the simulation environment by adding or removing the special objects.

There are two lists that the users can uses. By selecting an object at the first object list, the top one, the user is able to add new objects in the simulation environment at a certain desired place. The new added objects will also be visible at the second object list. This list contains all special objects that are present in the simulation environment. Using this list the user can remove the special objects from a certain place.

➢ Map

If we compare the map package carefully with the map component in the design architecture, one will notice that the implemented map package does not correspond to the map component in the design architecture. This issue can be explained that the implemented map package has two different functionalities. The first one is the representation of the world environment and the second one is the world that will be preloaded in the AIBO watchdog during the initialization stage. The difference between these two maps is that the last one only contains the static objects in the home environment and the first one also includes the special objects. This one is necessary to simulate the world environment. This environment simulation is required, because the lack of a proper object recognition program and sound recognition/localization program. Therefore the input of the environment needs to be simulated according to the available objects on the map. We have chosen to preload the environment in the AIBO watchdog during initialization of the AIBO. But only the static objects will be preloaded, so the special objects, e.g. fire, need still to be discovered during the patrolling of the AIBO watchdog. The simulation program will give the features of an object as input to the AIBO, when AIBO is standing next to that object. Figure 7.7 illustrates this situation.

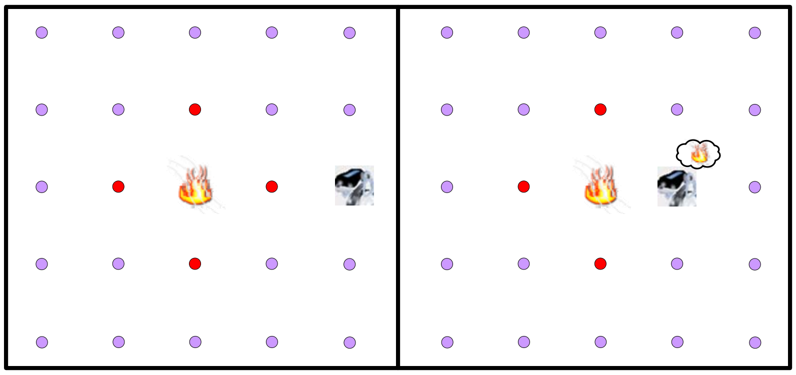


Figure 7.7: Illustration when AIBO will perceive the image features of an object. The red node represents the places where the AIBO is able to perceive the fire features by its camera. The left picture shows the start situation. The right picture shows the result of AIBO has reached that node.

The sound features will be passed to the AIBO, when AIBO is 2 distance units away from the sound source. If multiple sound sources are received at the same distance, the loudest one will be chosen. Figure 7.8 illustrates the situation that AIBO perceives the sound features of an object.

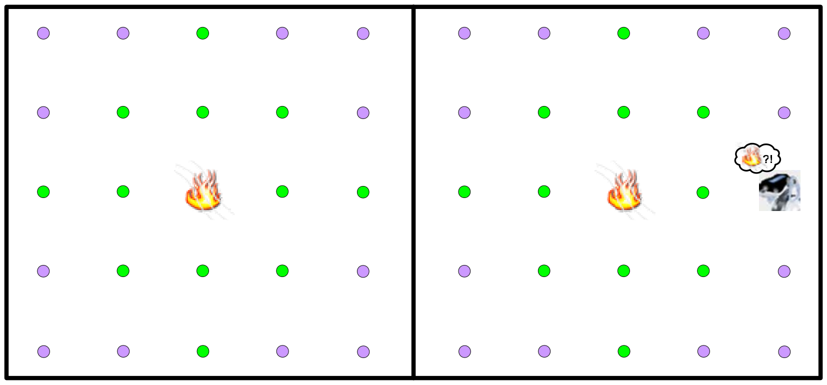
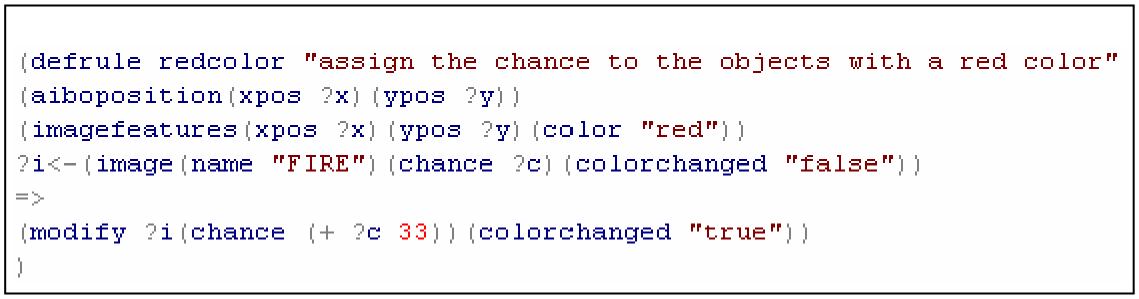


Figure 7.8: Situation when AIBO perceives the sound features of an object. The green nodes indicate that AIBO will perceive the sound features if AIBO reaches that node. At the left picture the start situation has been shown. The right picture shows the result when AIBO has reached that node.

➢ **Interpretation**

The package interpretation receives the features information of the simulation environment from the map package. These features information will be received every time when AIBO reaches a new waypoint. The received information will be matched with the object features in the expert system.

Objects can have image features and sound features. From images the color, shape and material will be extracted. The extraction will be processed by simulation. These features will be matched with the predefined object matching rules in the expert system. Figure 7.9 shows an example of a rule in the image object matching expert system.

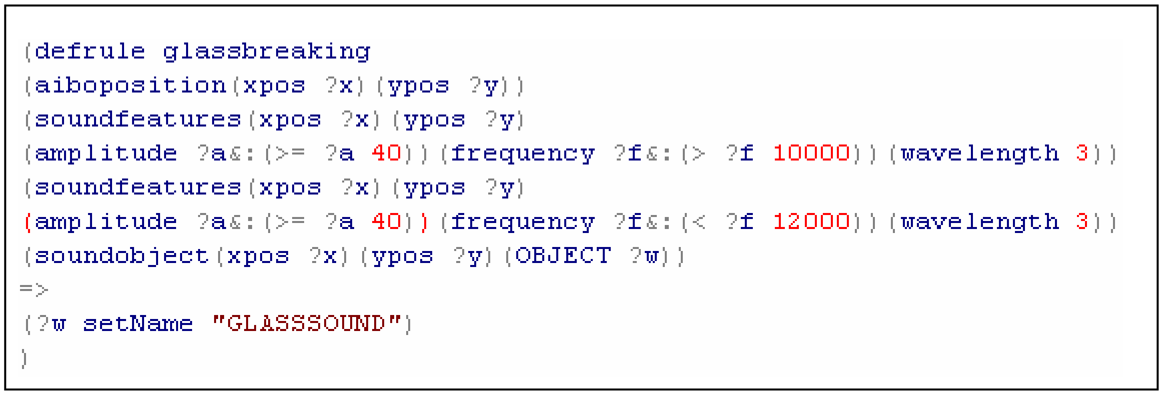


```
(defrule redcolor "assign the chance to the objects with a red color"
(aiboposition(xpos ?x) (ypos ?y))
(imagefeatures(xpos ?x) (ypos ?y) (color "red"))
?i<-(image(name "FIRE") (chance ?c) (colorchanged "false"))
=>
(modify ?i(chance (+ ?c 33)) (colorchanged "true"))
)
```

Figure 7.9: Example rule in the image object matching expert system. This rule assign a chance of 33 to the scenario FIRE, if a red color has been detected.

In this example a chance of 33 will be assigned to the fire object, if the red color is detected in the image. There are three different kinds of features to be matched with the matching rules in the expert system, so when all features are matching, it will result to a chance of 100% for that particular object. It is also possible to select an object, when there is no 100% match. By setting a threshold an object will be selected, when its chance is larger than the threshold.

From sounds the sound frequency, sound amplitude and wavelength will be extracted. In nearly the same way these features will be matched with the predefined sound object matching rules in the expert system. Figure 7.10 shows an example rule in the sound object matching expert system.



```
(defrule glassbreaking
(aiboposition(xpos ?x) (ypos ?y))
(soundfeatures(xpos ?x) (ypos ?y)
(amplitude ?a&:(>= ?a 40)) (frequency ?f&:(> ?f 10000)) (wavelength 3))
(soundfeatures(xpos ?x) (ypos ?y)
(amplitude ?a&:(>= ?a 40)) (frequency ?f&:(< ?f 12000)) (wavelength 3))
(soundobject(xpos ?x) (ypos ?y) (OBJECT ?w))
=>
(?w setName "GLASSSOUND")
)
```

Figure 7.10: Example rule in the sound object matching expert system. This rule will assign the glass sound to the sound object, when the amplitude is higher or equal to 40, frequency is between 10.000 and 12.000 and the wavelength is equal to 3.

As Figure 7.10 illustrates this expert system does not work with chances, but with 100% matching rules. This is implemented on purpose to show the flexibility and modularity of the designed architecture. The way, how the expert system works, can very easily be modified without having compatibility problems and other modifications in the designed architecture.

➢ **Reasoner**

The reasoner is the main part of the reasoning system and it determines the intelligence of the AIBO watchdog. The reasoner package consists of seven components which can be found in the designed architecture: manager, short term memory, long term memory, passive memory, scenario selector, scenario knowledge base system and scenario priority list component. Figure 7.11 shows the global class diagrams in the reasoner package.

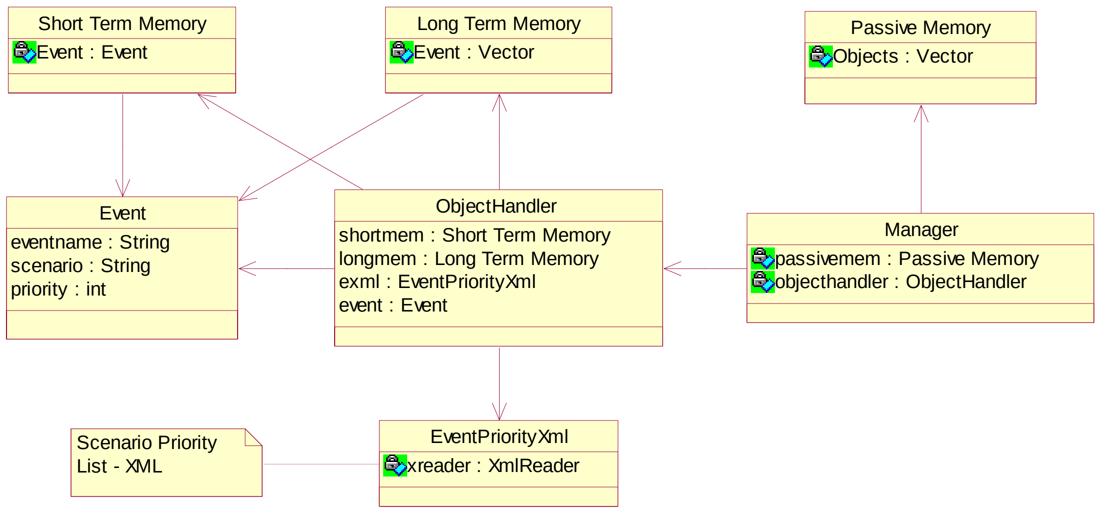


Figure 7.11: Global classdiagrams of the reasoner package. When a recognized object arrives at the Manager instance, it will be passed to passive memory instance and ObjectHandler instance. The ObjectHandler instance determines the current scenario based on that object.

The implementation of the memory components, short term memory, long term memory and passive memory are implemented according to the design of the architecture described in the design chapter. The long term memory and passive memory are holding a vector of event data and the short term memory is only holding one event. The class ObjectHandler represents the Scenario selector component and the scenario knowledge base system component presented in the designed architecture. The scenario knowledge base system is not implemented according to the design. Instead of reasoning with scenario probabilities a direct match will be executed. So the recognized object can only lead to one scenario. After matching the event the priority of the new scenario will be consulted. Depending on its priority it will be put in the Short Term memory or Long Term memory. The EvenPriorityXml class receives the priority inputs from an external XML file. This is conforming to the design of the scenario selection process.

➢ Eventhandler

The eventhandler package contains the classes that are needed to decide the reactions of the current scenario. This package contains 5 classes which represent the Event Handler component and Action Knowledge base system. Figure 7.12 shows the content of this package.

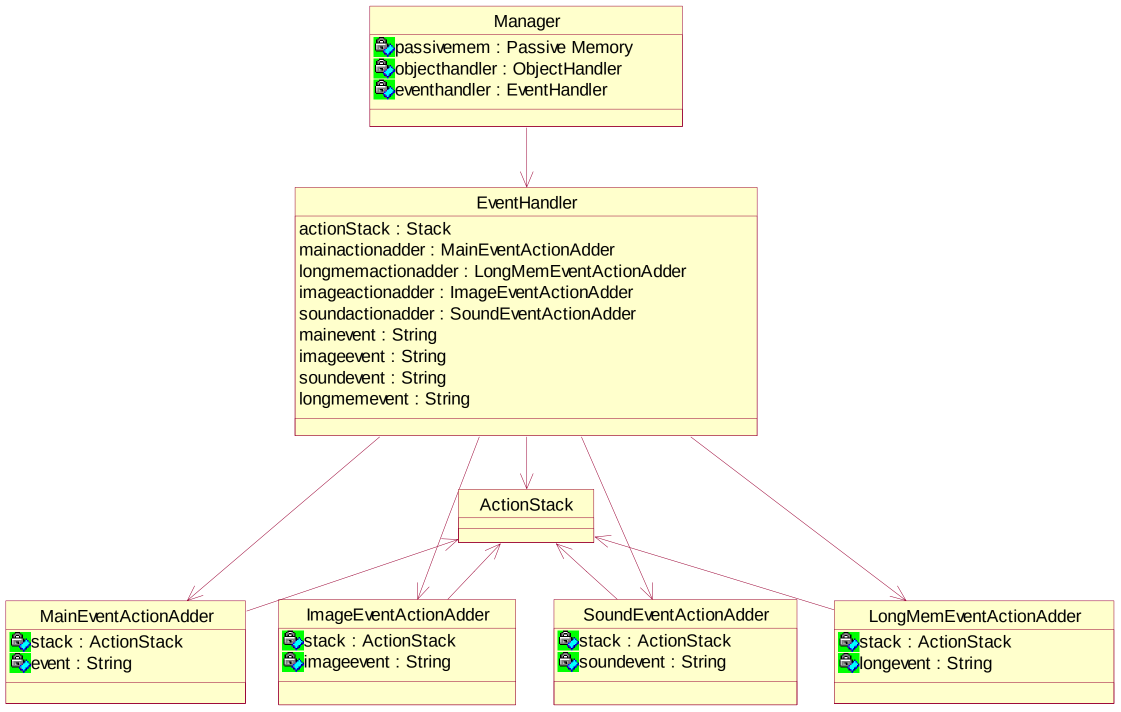


Figure 7.12: Classdiagrams of the eventhandler package. Based on the new recognized objects certain actions needs to be put on the stack. The 4 ActionAdder, expert systems, at the bottom determine the actions that need to be carried out.

The class EventHandler receives the scenario events from the class manager and this information will be passed to the rest of the 4 classes: MainEventActionAdder, ImageEventActionAdder, LongMemEventActionAdder and SoundEventActionAdder. These 4 classes represent the Action Knowledge base system component of the designed architecture. Based on the incoming scenario events the 4 classes will decide the actions with the predefined rules in its expert system. The differences between the classes ImageEventActionAdder, SoundEventActionAdder and MainEventActionAdder are the location aspect of the actions. ImageEventActionAdder and SoundEventActionAdder can only add actions that can be executed at the current position, e.g. capture picture or make sounds. This is needed, when a new scenario has been discovered which has a lower priority. In this situation the AIBO watchdog will not carry out the reactions of the new scenario, but it is also nice to take some evidence of the new scenario for the record. Therefore capturing the sound or picture will be used very frequently in these 2 classes. Figure 7.13 illustrates the working of this concept.

On the other hand the class MainEventActionHandler determines all reactions that are required to carry out to solve the current scenario, and put them on the stack. These actions can be technical operations, such as capturing the sound, but also physical actions, such as moving to a certain point. The last action can be used to explain the functionality of the class LongMemEventActionAdder. This class deals with the situation when the current scenario is completed and an old scenario from the long term memory need to be placed in the short term memory. In the past the technical operations, such as capturing the image has already been executed by the classes ImageEventActionAdder and SoundEventActionadder, therefore the AIBO only needs to move back to that point where the event has happened and investigate the situation again whether it has changed or not.

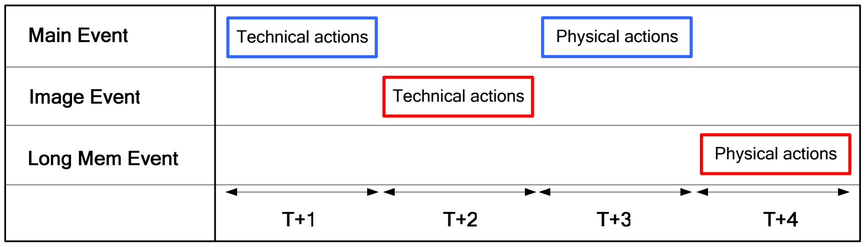


Figure 7.13: Timeline of actions that will be executed based on the origin of the event. Main Event will execute technical and physical actions. Image event only deals with technical actions. Long Mem Event from the long term memory to short term memory deals only with physical action.

➢ Execution

In the execution package one can find the classes that are required to execute an action. This package contains the following components from the design architecture: Action Stack and Executor component. The Actions Attributes XML component is not available. The actions attributes has been hard coded in the program. Figure 7.14 shows the global view of the classes inside the execution package.

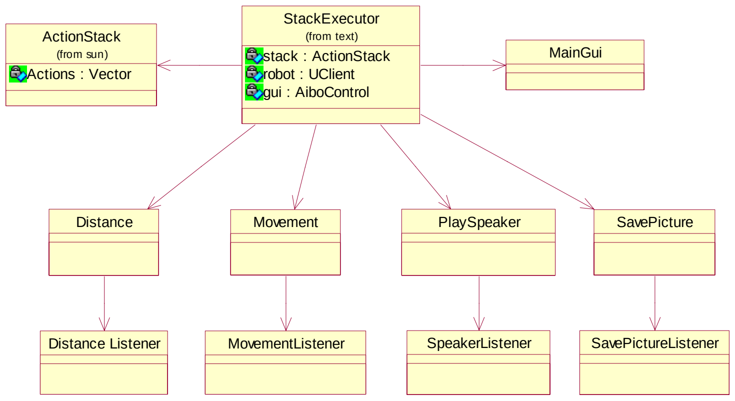


Figure 7.14: Classdiagrams of the execution package. The StackExecutor instance needs to execute the actions from the stack. The 8 classes at the bottom execute only a particular action of the stack. Every action that has been executed will be updated in the MainGUI.

The class StackExecutor obtains the next action from the stack and depending on the action it will pass that action to a certain action class. Every action that AIBO watchdog has realized, AIBO will send a message back to the action listener. As a result the program notices that the action has been taken place and the next action will be obtained from the stack. This process will last till all actions have been executed in the stack. After every action that is executed the visual representation of the AIBO environment and AIBO in the MainGUI will also be updated. Figure 7.15 shows a figure of the global total view of all implemented classes.

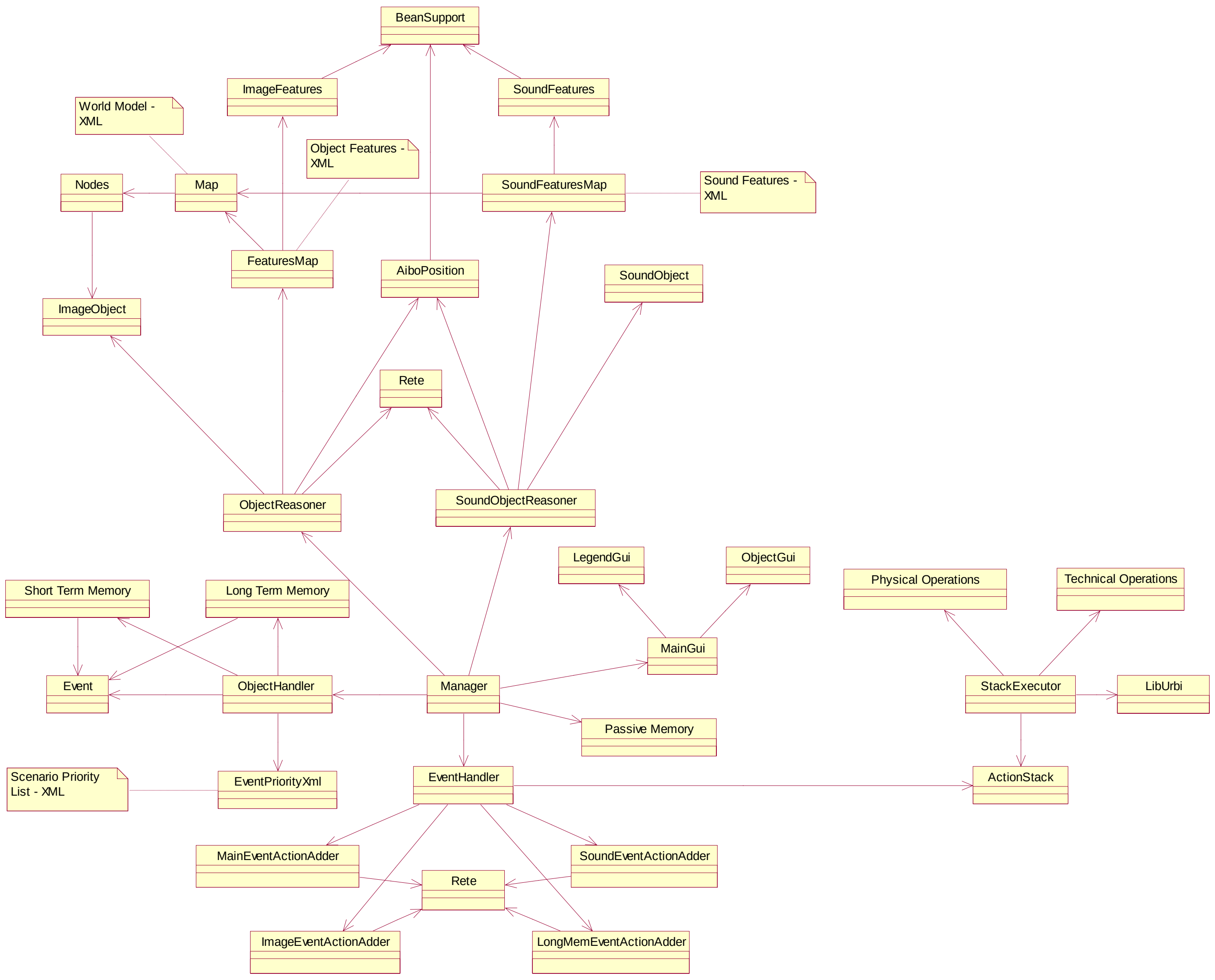


Figure 7.15: Global total view of all implemented class diagrams.

7.3. Manual

This section provides the knowledge to the user for using the program and modifying it to their desired settings. The first part of this section provides the knowledge to use the program and the second part of this section provides the knowledge to modify the settings to get the desired program.

7.3.1. Using the Program

- **Place the program on the AIBO**

The complete program consists of two parts. The first part of the program is running on the AIBO watchdog. This program let the external PC communicate with the internal operating system of the AIBO. It is based on the original libUrbi, but it has been modified to fulfill our program requirements. The content of the modified libUrbi program needs to be copied to the special pink colored Sony Memory Stick and be placed on the Sony Memory Stick slot in the AIBO.

- **Place the program on the PC**

The second part of the program will be run from the external PC which has installed the Java runtime environment (JRE) 1.4.2 and a proper Java compiler program which can execute a compiled java program. A recommended program is Eclipse.

- **Configure the settings of the wireless network card**

Before running the Java program confirms that the wireless connection on your PC has been set to ad-hoc mode and a fixed ip-address has been assigned to your wireless connection. The ip-address must be a version of 192.168.3.xxx where the xxx stands for a number from 0 to 255 except 14, because 192.168.3.14 is the ip-address of the AIBO watchdog. An example of an ip-address for the PC is 192.168.3.10.

- **Start the program on the PC**

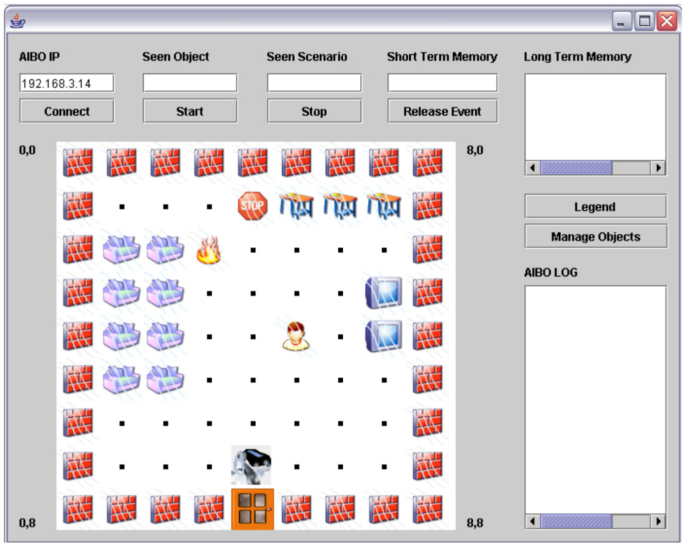
The Main method which starts the program is located in the java class manager. After compiling and running the program a Graphical User Interface will be prompted on the screen. Figure 7.16 shows the Graphical User Interface of the program.

- **Connect to the AIBO**

The standard ip-address of the AIBO has been set on the AIBO IP field, but we can change the value by clicking on it. After confirming the ip-address the user needs to click on the connect-button to make connection with the AIBO watchdog. When a connection with the AIBO has successfully been made, AIBO will produce a short sound and thereafter the lights on its back will be lightened up. This is a signal that the initialization process of the AIBO watchdog has successfully been executed.

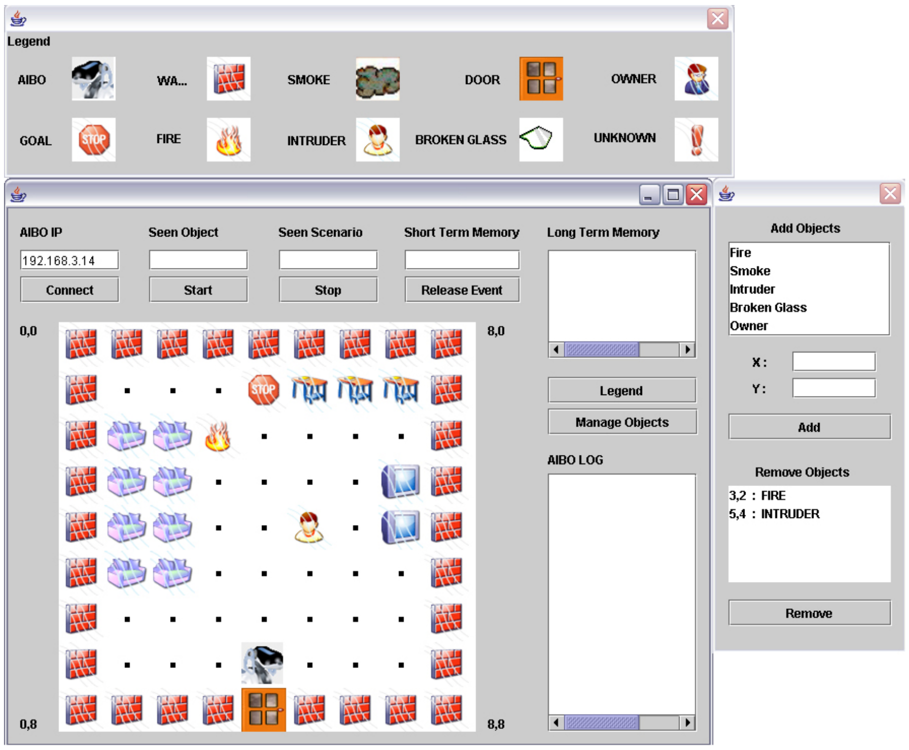
- **Show the extra GUI’s**

By clicking the legend button and manage objects button new dialogues will be prompted on the screen. Figure 7.17 illustrates this situation. The top GUI is the LegendGUI where icons representation can be found. The GUI in the middle is the MainGUI. The states of the AIBO watchdog can be read from the GUI. The right GUI is the ObjectGUI. From this GUI objects can be added or removed.



AIBO IP: 192.168.3.14

Figure 7.16: The User Interface after initialization of the program.



AIBO IP: 192.168.3.14

Add Objects:
Fire
Smoke
Intruder
Broken Glass
Owner

Remove Objects:
3,2 : FIRE
5,4 : INTRUDER

Figure 7.17: All GUI's in the program. The top GUI is the LegendGUI where icons representation can be found. The GUI in the middle is the MainGUI. The states of the AIBO can be read from the GUI. The right GUI is the ObjectGUI. Objects can be added or removed.

- **Add or remove objects**

New special objects can be added using the ObjectGUI. For adding new objects the user needs to select one of the options at the object list and specify the x-coordinate and y-coordinate values. By clicking on the add button an object is added in the simulation. The new object can be found at the remove object list and also on the picture of the home environment. To remove an object the user needs to select an object from the remove object list and click on the remove button. The selected object will disappear from the remove object list and also on the picture of the home environment.

- **Start the AIBO watchdog**

After setting up the environment with the desired objects the user can start the patrolling process of the AIBO by clicking on the start button. AIBO will now patrol the environment and search for the abnormal objects in its environment. All physical movements of the AIBO can be followed by looking at the picture on the MainGUI. This picture is a real-time representation of the AIBO location and its environment.

- **Encounter special objects**

Every time when AIBO encounters a special object the name of the object will be displayed on the seen object field and the corresponding scenario will be displayed on the seen scenario field. This scenario will also be displayed on the short term memory field or the long term memory text area.

- **See the patrolling history**

During the patrolling of the AIBO the detected objects will be displayed on the AIBO log text area. Not only will the object name be displayed, but also the location of the discovered object. As a result the user can track the object history that AIBO has encountered.

- **Stop AIBO**

The patrolling procedure can be stopped by either closing the program or clicking on the stop button. The AIBO will stop executing actions immediately. This is useful when you do not want the AIBO running in your way. The stop button has not been implemented yet. Therefore closing the program is the only option to stop execution.

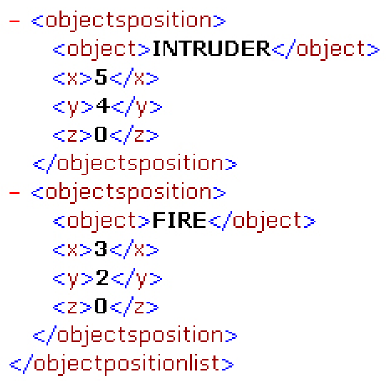
- **Continue patrolling after completing a scenario**

Every time when AIBO has completed a scenario, it will stay at the goal destination and making sounds. AIBO will stay like this till someone will free it from that state. For example when AIBO has run to the main door because of a fire, it will wait till the fire scenario has finished. The fireman has to master the fire before the AIBO will go in patrolling again. By clicking on the release event button the user gives the AIBO a signal that the current scenario has already been completed. In the future the user can also touch the back sensors of the AIBO to complete a scenario. After completing the current scenario the AIBO will continue patrolling to find a new one or completing the ones in the long term memory.

7.3.2. Modifying the Settings

➢ Modify AIBO environment

The start configuration of the AIBO environment is described in the world model XML file, objectposition.xml. This file will be loaded into the program when AIBO starts up. By changing this XML file the user is able to change the start environment of the AIBO watchdog. Figure 7.18 shows a part of this XML file.



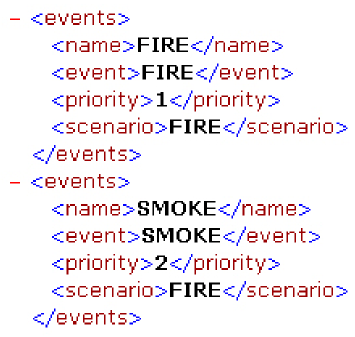
```
- <objectsposition>
    <object>INTRUDER</object>
    <x>5</x>
    <y>4</y>
    <z>0</z>
  </objectsposition>
- <objectsposition>
    <object>FIRE</object>
    <x>3</x>
    <y>2</y>
    <z>0</z>
  </objectsposition>
</objectpositionlist>
```

Figure 7.18: Part of the world model xml file. The object tag gives the name of the object. The x, y and z tags are respectively the x, y and z coordinates of the objects location.

An object has 4 attributes: name of object, x-coordinate value, y-coordinate value and z-coordinate value. By modifying these attributes the kind of object and its location will be changed. Inserting new objects can be performed by adding the complete objectsposition tag. Removing objects from the environment can be performed by removing the complete objectsposition tag. As a result the desired environment can be created.

➢ Modify Scenario Priority list

The priority of scenarios is described in the scenario priority list XML file, events_priority.xml. Modifying this file the priority of scenario can be adapted. Figure 7.19 illustrates a part of the scenario priority list XML file.



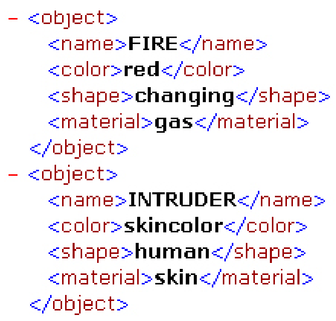
```
- <events>
    <name>FIRE</name>
    <event>FIRE</event>
    <priority>1</priority>
    <scenario>FIRE</scenario>
  </events>
- <events>
    <name>SMOKE</name>
    <event>SMOKE</event>
    <priority>2</priority>
    <scenario>FIRE</scenario>
  </events>
```

Figure 7.19: Part of the scenario priority list XML file. The name tag needs to be matched with the recognized object. The event, priority and scenario tags will assign the value to the recognized object.

An event has 4 attributes: name, event, priority and scenario. The tag name indicates the name of the incoming object and the tag event specifies the name of the event. Furthermore the tag priority specifies the priority of the scenario. The lower the number the more important it is. At last the tag scenario specifies the name of the scenario. Adapting the priority can be performed by changing its priority number. Inserting new events can be completed by adding the tags between the events tags and removing the object by removing the tags between the events tags. Procedure is the same as the AIBO environment XML.

- **Modify the simulation objects attributes**

From the simulation environment the objects will be converted to objects attributes which the AIBO is able to sense. By passing these attributes to the interpretation component it will try to recognize the original object. The conversion of the objects to the objects attributes will be performed by simulation based on the object attributes XML file, objectfeatureslist.xml. Figure 7.20 shows a part of the object attributes XML file.



```
- <object>
    <name>FIRE</name>
    <color>red</color>
    <shape>changing</shape>
    <material>gas</material>
  </object>
- <object>
    <name>INTRUDER</name>
    <color>skincolor</color>
    <shape>human</shape>
    <material>skin</material>
  </object>
```

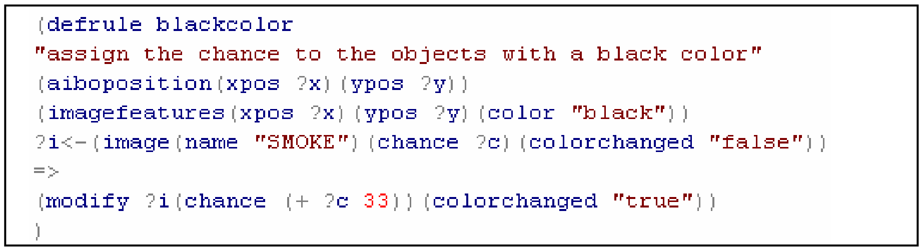
Figure 7.20: Part of the object attributes XML file. The objects in the simulation environment will be simulated with the values in the color, shape and material tags.

An object has 4 attributes: name, color, shape and material. In this simulation only the last 3 attributes are used to match the corresponding object. The user can change these 3 attributes to redefine the object attributes. Inserting an object can be completed by inserting these 4 tags at the end of the file. Removing the complete codes between the object tag leads to a removal of an object.

- **Modify the interpretation process**

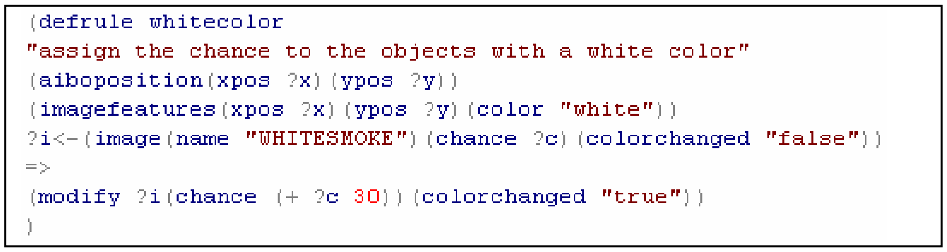
The interpretation process is performed by an expert system. The file ObjectFeaturesList.clp is the corresponding expert system. This system contains a lot of rules which are used to recognize the corresponding object. Figure 7.21 shows a rule from the expert system.

The assigning of chances to the objects takes place when a certain objects attribute matches. In this case the color black raises the chance of smoke by 33. New rules can be inserted by copying one of the rules and change the values of the attributes. For example the user wants to insert a new rule for white smoke. If the color white is detected, it will raise the chance of white smoke by 30. Figure 7.22 shows the result of the adjustment.



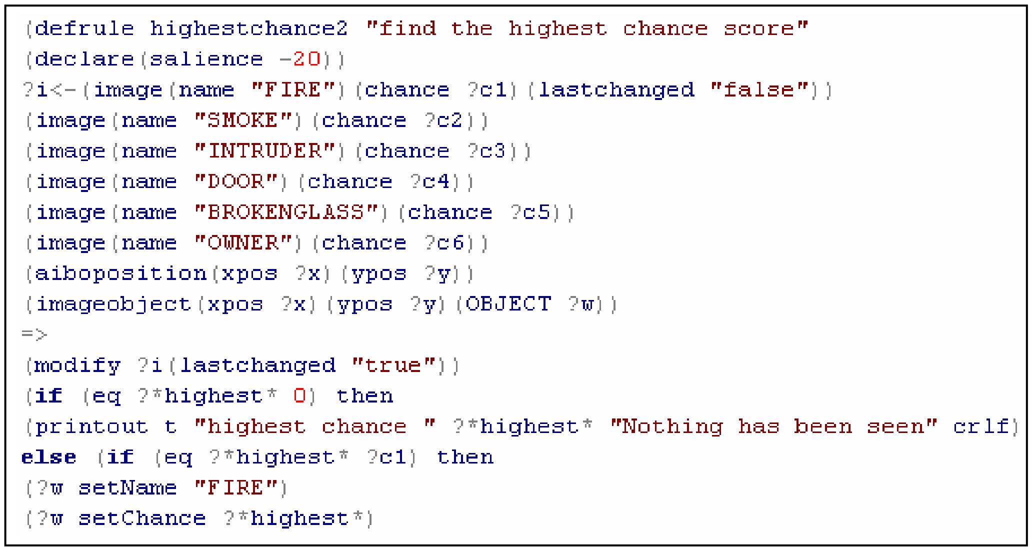
```
(defrule blackcolor
"assign the chance to the objects with a black color"
(aiboposition(xpos ?x)(ypos ?y))
(imagefeatures(xpos ?x)(ypos ?y)(color "black"))
?i<-(image(name "SMOKE")(chance ?c)(colorchanged "false"))
=>
(modify ?i(chance (+ ?c 33))(colorchanged "true"))
)
```

Figure 7.21: One of the rules in the expert system. This rule raises the chance of the smoke object by 33 when a black color has been detected.



```
(defrule whitecolor
"assign the chance to the objects with a white color"
(aiboposition(xpos ?x)(ypos ?y))
(imagefeatures(xpos ?x)(ypos ?y)(color "white"))
?i<-(image(name "WHITESMOKE")(chance ?c)(colorchanged "false"))
=>
(modify ?i(chance (+ ?c 30))(colorchanged "true"))
)
```

Figure 7.22: The new inserted rule in the expert system. This rule raises the chance of WhiteSmoke object by 30 when the white color has been detected. After assigning the chances the program needs to find the object with the highest chances.



```
(defrule highestchance2 "find the highest chance score"
(declare(salience -20))
?i<-(image(name "FIRE")(chance ?c1)(lastchanged "false"))
(image(name "SMOKE")(chance ?c2))
(image(name "INTRUDER")(chance ?c3))
(image(name "DOOR")(chance ?c4))
(image(name "BROKENGLASS")(chance ?c5))
(image(name "OWNER")(chance ?c6))
(aiboposition(xpos ?x)(ypos ?y))
(imageobject(xpos ?x)(ypos ?y)(OBJECT ?w))
=>
(modify ?i(lastchanged "true"))
(if (eq ?*highest* 0) then
(printout t "highest chance " ?*highest* "Nothing has been seen" crlf)
else (if (eq ?*highest* ?c1) then
(?w setName "FIRE")
(?w setChance ?*highest*)
```

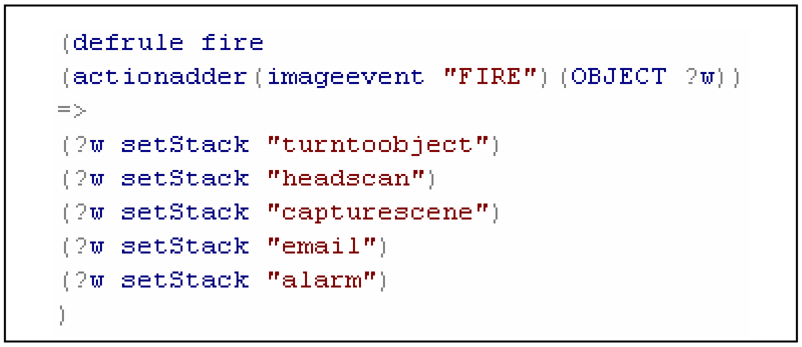
Figure 7.23: A part of the rules in the expert system to find the highest chance.

Figure 7.23 illustrates this process in the expert system. All algorithms in this expert system can be changed to the desired ones. New variables and java classes can be introduced. To make it compatible with other classes it is necessary to understand that the final result will be assigned to the instance imageobject. The new name and chance attribute of the instance imageobject will be assigned. The inputs at the start for matching are the instances aiboposition and imagefeatures. These 2 instances are used to find the final result. Any other instances that are needed have to be imported by the user.

➢ Modify the actions allocations of the corresponding scenario

Depending on the current scenario reactions will be determined and put on the stack. The determining of the actions will be processed by the expert system, maineventhandler.clp. Figure 7.24 shows an example rule in the expert system which will put the actions on the stack.

Figure 7.24 shows a rule which put actions in the stack, when a fire event has been detected. Actionadder is an instance of the stack. Using the setStack method new actions can be put on the stack. The assigned sequence in the rule is also the sequence of execution. The last action that is put on the stack will also be executed as the last one. More information about the kinds of reactions can be found in Appendix C.



```
(defrule fire
(actionadder(imageevent "FIRE")(OBJECT ?w))
=>
(?w setStack "turntoobject")
(?w setStack "headscan")
(?w setStack "capturescene")
(?w setStack "email")
(?w setStack "alarm")
)
```

Figure 7.24: Example rule in the ActionAdder’s expert system. If the current scenario is FIRE, then the actions listed with setStack will be put on the action stack.

➢ Modify the icons

The used icons of the Graphical User Interface are located at the /icons directory of the program. Icons can easily be replaced by another one. Replacing the icons can be completed by copying the new icons in the same directory and by giving the same name. To let the new icons fit in the User Interface it is necessary to keep the size of the icon at 44 x 44 pixels. The representing picture must be in the middle of the icon. By changing the code in the java class LegendGui one is able to change the settings of the icons.

8 Experiments & Results

“Gold is tested by fire, software by use.”
Bou Tsing Hau

In this chapter the experiments results of the implemented framework will be presented. First an explanation will be given about the used test methodology and thereafter the results of the test experiments. Last but not least a conclusion will be derived from the test results. These test experiments were demonstrated during a special meeting of the MMI department at the faculty EEMCS of TUDelft.

8.1. Test Methodology

There are 4 test scenarios which are designed to test the implemented architecture. At each test scenario a description about the scenario will be provided and also the reactions that AIBO should execute when it encounters that particular object. The result of the test scenarios will be provided at the end of each test scenario. In order to test the implemented architecture the entire home environment in the simulation has been replicated in a real environment. This way it can show that the implemented framework does not only work in the simulation mode on the PC, but also in practice.

These 4 designed test scenarios are able to show the possibilities of the designed reasoning system. There can be single events occurring in a scenario or multiple events. Multiple events can occur sequentially or in parallel. Both situations will be presented in the 4 designed scenarios. Sudden changes on the map, such as sudden appearing of unknown obstacles, must also be avoided by the AIBO watchdog. This situation will also be presented in the 4 designed scenarios. The default map of the home environment which is given to the AIBO beforehand is illustrated by Figure 8.1. This home environment has been replicated in the real environment. Paper boxes are used to represent the objects in the simulation. In order to represent the special objects, such as intruder and fire, some toys were used. The floor where the demonstration was located is very smooth. If another floor is chosen, there is a probability that the AIBO movements will not function properly.

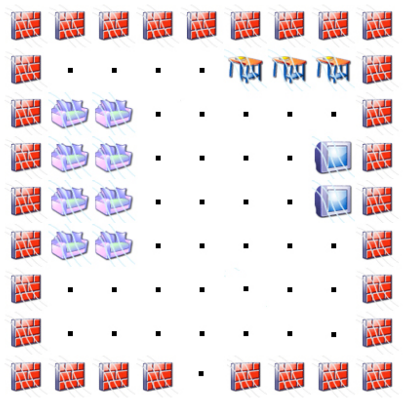


Figure 8.1: A predefined map that will be loaded during the initialization process of the AIBO. Only the static objects are loaded in the AIBO watchdog.

Using the preloaded map AIBO is able to maneuver from one point to another point in its environment. AIBO will react on the special objects which are not present in this map. The reaction behavior of AIBO on a certain object has already been described in the design chapter, chapter 4. The specifications of the test environment where the test scenarios where conducted is illustrated in Table 8.1

Table 8.1: Test environment specifications

Test Environment	Specifications
AIBO	AIBO-ERS7
Place	A room with a smooth floor (no carpet).
Objects	Objects are represented by paper boxes with an icon.
Special Objects	Special objects are represented by toys.
Size environment	About 2 by 2 meters. The distance between each node is around 20cm.
External PC	A Centrino laptop with a wireless connection, IEEE 802.11b/g.

8.1.1. Scenario 1: Intrusion with Broken Glasses

This scenario describes a situation where two different events occur sequentially and the default handling method will be used. There are no unknown obstacles introduced in this scenario. A short summary about this test scenario is presented in Table 8.2

On a cloudy day AIBO is patrolling the house environment. At this moment nothing suspicious are found. Suddenly AIBO hears some glass sound and runs to that direction. On the floor AIBO sees some broken glasses and decides to explore the entire home environment to find the cause. During exploration AIBO hears some human voices and under careful inspection AIBO discovers an intruder. AIBO takes a picture and barks at the intruder. Thereafter AIBO is going to a certain place where it usually can find his owner and create a big alarm sound.

Table 8.2: Test scenario 1 specifications

	Specifications
Events	Broken glass event and intruder event. (Sequential)
Solution	Default in program
Obstacles	None

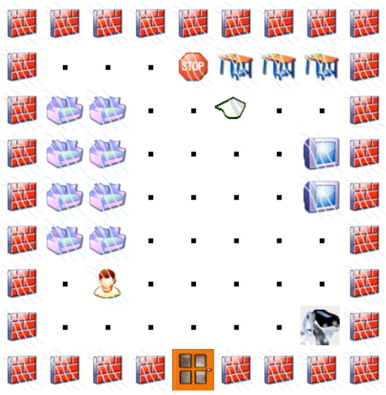


Figure 8.2: Map representation of the start environment of the first scenario. AIBO is located at the right bottom and it has to navigate to the red stop sign. Intruder and broken glasses are introduced in this scenario.

Figure 8.2 shows the start situation of the home environment. AIBO should walk to the goal destination and discover the broken glasses. During handling the scenario of the broken glasses AIBO should discover the intruder and react on it. By finding the intruder the broken glass scenario has been completed. The reactions on the intruder scenario will be executed now. Figure 8.3 shows the result after handling the intruder scenario.

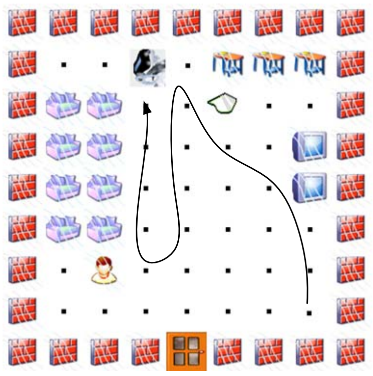


Figure 8.3: The result of the first test scenario. The curved line shows the path that AIBO has taken.

Experiment Results:

During the experiments the reasoning system of the AIBO has performed as expected. The reactions of AIBO watchdog was executed exactly as expected. The intruder scenario was executed according to the description, described above. There were no mistakes that AIBO has made, but the movements of AIBO are not accurate enough. After turning left and right AIBO does not face the original direction, but it faces a little bit more to the right. This turning algorithm was part of the libUrbi program, but after adjusting the parameters it is still not perfect. Therefore it was needed to correct the AIBO location before AIBO is going to walk a long distance. After clicking on the release event button, when AIBO has completed the intruder scenario, the AIBO watchdog was continuing to patrol the home environment.

8.1.2. Scenario 2: Escape Caused by Fire.

This scenario describes a situation that AIBO has discovered a useful object which will be used for the solution of the detected event. There are no unknown obstacles introduced in this scenario. Table 8.3 summarizes the environment specifications.

Table 8.3: Test scenario 2 specifications.

	Specifications
Events	Fire event
Solution	Use door object, which has been found during patrolling.
Obstacles	None

In a peaceful afternoon AIBO is patrolling the home environment. Its owner has just left for his work. During its patrolling AIBO discovered the main door. After a while AIBO sees some black smoke in the direction of the table. When it steps into the smoke area it hears the sound of fire and sees that the flames are growing. AIBO stays calm, takes a picture of this scene and create loud sounds. AIBO is now thinking how to handle this situation. Because AIBO has detected the main door during its patrolling and it knows that the door can be used to escape, AIBO is running to the main door and create a big alarm to notify the people in its environment.

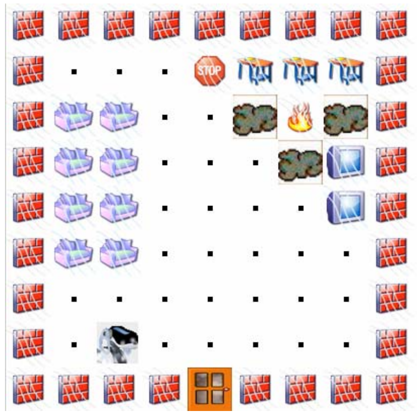


Figure 8.4: Map representation of the start environment of the second scenario. AIBO is located at bottom left and it is heading to the red stop sign. In this scenario the fire, smoke and door objects are introduced.

Figure 8.4 shows the start environment of this scenario. AIBO will walk pass the door and thereafter to the goal destination. The door has a special meaning for the AIBO, because it can be used for escaping in emergency situations. When AIBO almost gets to the destination, it detects the smoke and fire and after reasoning it will go back to the door. Figure 8.5 illustrates these movements. This scenario involves the using of a useful object that AIBO has encountered during patrolling.

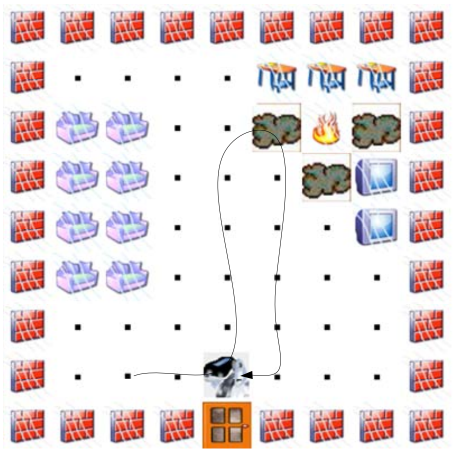


Figure 8.5: The result of the second test scenario. The curved line shows the path that the AIBO has taken. AIBO is walking back to the door to escape this environment, after it has detected the fire.

Experiment results:

The experiment results have proved again that the reasoning system of the AIBO is working correctly. All steps that AIBO has taken can be explained before the AIBO starts. The passive memory component is working properly. Only the movements of AIBO can be improved. Therefore it was again necessary to correct the location of the AIBO before AIBO is going to walk from the fire destination to the door.

8.1.3. Scenario 3: Behavior Triggered by Ringing Doorbell

This scenario describes a situation that AIBO encounters a useful object which will be used to solve the discovered event. Sudden obstacles are present to hinder AIBO. AIBO needs to find another path to walk to its destination. Table 8.4 summarizes the environment specifications.

Table 8.4: Test scenario 3 specifications.

	Specifications
Events:	Ringing door bell event.
Solution:	Use owner object, which has been found during patrolling.
Obstacles:	Yes

In a very quiet evening the AIBO owner is sitting on the sofa and watching an action movie. AIBO is very hard working, so it is still patrolling the home environment when its owner is watching a movie. Suddenly when AIBO passed by the door it hears the ringing sound of the bells. Someone is standing behind the door. Since AIBO is not able to open a traditional door AIBO walks to its owner and create a requesting sound to notify the owner that someone has rung the bells. During its way back to the owner, some not foreseen obstacles are blocking the fastest path. Therefore after detection by AIBO it needs to find another path to notify its owner.

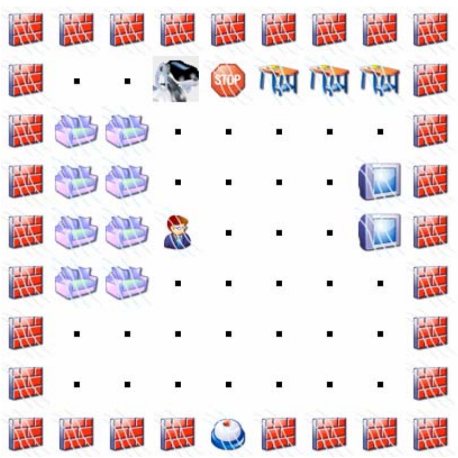


Figure 8.6: Map representation of the start environment of the third scenario. AIBO will navigate to the red stop sign and calculate its next goal. This scenario introduces the owner object and the ringing bell.

Figure 8.6 shows the start environment of this scenario. After reaching the destination point a new destination point at the left bottom will be created. During patrolling to the new destination point AIBO discovers its owner and thereafter the ringing of the doorbell. After hearing the doorbell AIBO will walk back to notify its owner. When AIBO is walking back, an object will be placed on the path that AIBO is planning to take. AIBO needs to detect this object and calculate the new path to its owner. After arriving the goal destination AIBO will create an alarm sound. Figure 8.7 illustrates the movements of the AIBO.

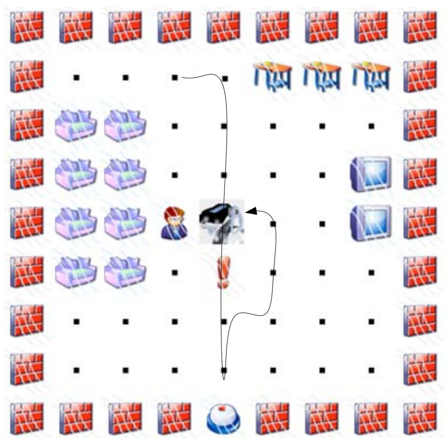


Figure 8.7: The result of the third scenario. The curved line shows the path that AIBO has taken during the third test scenario. The exclamation mark shows the unknown object that was detected during its navigation to its owner. Therefore another path needs to be taken.

Experiment results:

The only difference between this scenario and scenario 2 was the sudden blockade. In this situation AIBO needed to recalculate the new path that it had to take. After detecting the sudden blockade the reasoning system of the AIBO watchdog was still producing the correct reactions. The AIBO watchdog went around the blockade to reach the destination point and it executed the actions that were necessary. The movement issues were still present in this scenario. Correcting the location of the AIBO was the only option.

8.1.4. Scenario 4: Intruder who has lighted up the Fire

This scenario describes a situation where AIBO has 2 different events in his mind. The most important one will be executed first. After completing the first event the second event will be executed. During execution of the first event there are obstacles on the AIBO path, therefore AIBO needs to find another path to avoid collision. Table 8.5 summarizes the environment specifications.

Table 8.5: Test scenario 4 specifications.

	Specifications
Events:	Intruder event and fire event. (Parallel)
Solution:	Use a door object, which has been found during patrolling.
Obstacles:	Yes

On a sunny day AIBO is walking around in a messy home environment, but nothing suspicious has been found yet, except it encounters the main door during patrolling and some unknown objects on the path. Suddenly AIBO hears some human sounds and it walks to that area for inspection. AIBO detects an intruder and before it goes to the certain point to alarm the owner, it takes some pictures and creates some loud sounds. While AIBO is walking to that certain point, AIBO hears the sound of flames and detects the smoke and fire. After taking pictures and creating some loud sounds AIBO is now walking to the main door to alarm the people. Because of the mess in the environment many obstacles have blocked the AIBO path. Therefore it takes a while, before AIBO reaches the main door. The intruder event will be ignored for a while till the fire event has been handled.

Figure 8.8 illustrates the start situation of the fourth scenario. AIBO starts at the door location which will be stored in the passive memory. When AIBO is going northwards it detects the intruder. A picture of the intruder will be captured and alarm sounds will be created. AIBO is now heading to the certain place where it usually can find its owner. While AIBO is walking to that certain place, AIBO has detected the burning fire. To protect itself AIBO is running back to the main door and alarm the people in the outside environment. When AIBO is walking back to the main door, an obstacle has blocked its path. Therefore a new path will be calculated and taken. Figure 8.9 illustrates this process.

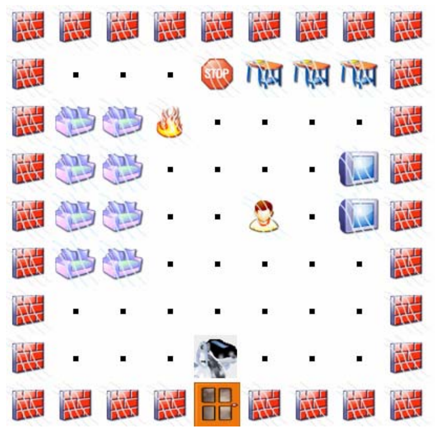


Figure 8.8: Map representation of the start environment of the fourth scenario. AIBO will detect the fire and intruder object.

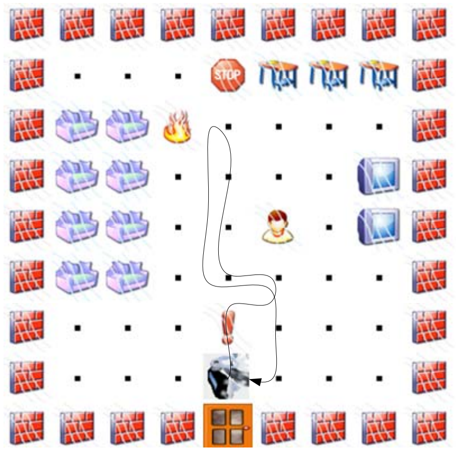


Figure 8.9: The result of the fourth test scenario. The curved line shows the path that AIBO has taken. The exclamation mark shows the unknown object that suddenly appeared in front of AIBO when it was heading the door.

Experiments results

AIBO has prioritized the two parallel occurring scenarios and solving them one by one. The scenario with the highest priority was solved first and thereafter the second highest. This prioritizing approach was functioning as explained in the design chapter, chapter 4. The implemented memories were working properly. The data switching from long term memory to short term memory could be followed from the MainGUI. The navigator has calculated another path, when an obstacle was blocking the current path. The negative side of the results is the physical navigation of the AIBO. Because of its inaccurate physical movements, it was necessary to correct the position of the AIBO during the experiment, but this has already been mentioned at the previous experiments.

8.2. Test Conclusion

From the experiments results of the 4 test scenarios we can conclude that during the test scenarios the working of AIBO watchdog was performing as expected. The incoming features were simulated by the program and these features were interpreted by the expert system. From this point the reasoning of the recognized objects started. The manager passes the incoming information to the destined parts of the reasoning system. The 3 memory components were working as expected. The useful objects, e.g. door and owner, were stored properly in the passive memory and they were used when needed. Prioritizing the incoming scenarios was also successful. The most important scenario was stored in the short term memory and the other incoming scenarios were stored in the long term memory. When the current scenario was completed successfully the last scenario from the long term memory was placed in the short memory. This scenario switching in the memory components was working properly.

The corresponding actions of the scenario were determined properly by the expert system and these actions were executed one by one. Obstacles were avoided and a new path was calculated. The state of the environment and AIBO position was updated correctly in the picture of the MainGUI. From the MainGUI the reasoning state of the AIBO could be followed. New objects that were added during the simulation were detected properly by the AIBO. The only problem, that was hindering the AIBO watchdog, was its physical movements. The implemented program requires that the AIBO watchdog can turn and walk accurately, but in practice AIBO is having difficulties with turning and walking. AIBO does not turn 90 degrees to the left or right, but 80 degrees to the left and 85 degrees to the right. Depending on the paw positions the walk algorithm will not always walk straight. If both left paws are between the right paws, AIBO will have a deviation to the left. Figure 8.10 illustrates this situation.

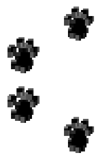


Figure 8.10: A situation which leads to incorrect walking movements of the AIBO watchdog. The positions of the left paws are between the right paws.

9 Conclusion and Recommendations

"The best way to predict the future is to invent it."
Alan Kay

As the result of the continuous increasing of violence worldwide it is needed to increase the surveillance for the safety of people. With this project we have tried to achieve a robot dog which is able to look after a home environment. AIBO is a multifunctional robot dog. Because of these integrated functionalities it is a good research medium to understand more about robots. This project has brought us a closer look at the robot world. The day that robots can help us guard the home environment will just be a matter of time. In this chapter the results of the project will be assessed against the requirements that were defined at the start and the results of the implemented architecture of the AIBO watchdog will be discussed. Furthermore recommendations for future work will be given.

9.1. Conclusion

This thesis report presents a research in developing an AIBO watchdog that is able to safeguard a home environment. The architecture and the reasoning system of AIBO watchdog was the main focus of this project. The resulting AIBO watchdog is able to process information from the environment and generate appropriate actions to take care of the home environment.

More specifically, the developed AIBO watchdog is able to process features of simulated objects in the environment and recognize the corresponding objects. Based on this information AIBO watchdog creates scenario hypotheses and tries to understand the current situation. When multiple hypotheses are occurring at the same time AIBO is able to prioritize these hypotheses to select the most important scenario. The most important one gets the attention of the AIBO watchdog and the other scenarios are dealt with afterwards. Based on the selected scenario the AIBO watchdog selects the corresponding reaction and these reactions will be executed sequentially. Possible reactions could be walk forward, walk backward, bark, etc. Some basic reactions such as walk forward and walk backward were already available for the URBI framework, used for the AIBO watchdog. So these could be used directly, some reactions had to be modified and others had to be completely implemented from scratch.

The developed AIBO watchdog has been tested in a test environment that consisted of some small objects and paper boxes that represented certain objects in the simulation. Within these simplifications the test results have confirmed that AIBO watchdog is able to take care of a home environment. Following is a point by point assessment of the thesis requirements.

9.1.1. Design an Architecture

The AIBO watchdog was almost designed from scratch. The work of the predecessor of the AIBO watchdog project was only used to have an idea of the working of AIBO. The interactions of AIBO and its environment need to be described and an appropriate framework needs to be designed. The first requirement:

- ***Requirement 1:*** Design an architecture to let the sensors detection component, reasoning component and action component collaborate efficiently with each other.

has been fulfilled in chapter 5. A new architecture was designed to let the AIBO watchdog interact efficiently with its environment. This architecture is based on the popular agent architecture. By conducting literature surveys on currently existing agent architectures the new architecture has been developed. The developed architecture is modular and flexible, therefore when applying this watchdog functionally on robots other than the AIBO robot very little modifications are needed. Not all parts of the designed architecture have been implemented, only the parts that are required to show the correct behavior of the AIBO watchdog. These parts have been tested extensively.

9.1.2. Design a Reasoning System

The main focus was the reasoning system of the AIBO watchdog. Based on the incoming inputs AIBO understands its current situation and makes an intelligent judgment. The corresponding reactions are developed and executed by the AIBO watchdog. Therefore, the second requirement:

- ***Requirement 2:*** *Design an intelligent reasoning component which is capable to collaborate with the sensors detection component and action component.*

has been achieved. The developed reasoning system is based on dynamic scripting that uses threshold and ranking. Based on the incoming inputs the most plausible scenario is calculated. The scenarios with probabilities bigger than a fixed threshold will be ranked in priority. The most plausible scenario is the scenario that is at the first place of the rank. Thereafter reactions that belong to that scenario are determined and executed. The developed reasoning component was completely designed, but partly implemented. This reasoning component has been tested extensively during the conducted experiments.

9.1.3. Design World Model and Corresponding Navigation Algorithm

AIBO is able to move around and explore its environment. As a result AIBO is able to observe related events. A description about the world environment of the AIBO watchdog has been made.

Based on the world environment a corresponding navigation algorithm has been developed. Therefore, the third requirement:

- ➢ ***Requirement 3:*** Design the world model of the AIBO watchdog and its corresponding navigation algorithm.

has been achieved. The world model of the AIBO watchdog contains objects, functionalities and their relationship. For the navigation algorithm the world environment is divided into a grid. The intersection of the vertical and horizontal grid lines are the waypoints. At each waypoint there are 4 options: walk forward, walk backwards, turn left and turn right. AIBO is able to move from one waypoint to the other waypoints. Using this waypoint navigation algorithm AIBO watchdog is able to move to most of the places and patrol the environment.

9.1.4. Implement a Prototype

To prove that the designed architecture could work properly, the AIBO watchdog has been implemented. The reactions of AIBO are not only presented in the software, but they are also executed by the real AIBO watchdog. This leads to the fourth requirement of the AIBO watchdog project:

- ➢ ***Requirement 4:*** Implement a prototype which can prove the proper working of the designed architecture and reasoning component, in the designed world model.

Most of the components in the designed architecture were implemented. The implemented program was able to demonstrate the correctness of the reasoning system. Each step of the reasoning process can be followed by the GUI of the program. The output of the reasoning process leads to physical reactions of the AIBO watchdog. The entire implementation has been explained in very detail and a manual has also been written to let people adapt the software to their own version. Therefore this requirement has also been achieved.

9.1.5. Test Prototype

In chapter 8 the results of the implemented architecture has been presented. Experiment with test scenarios has been conducted and demonstrated during a special meeting. This experiment has shown the correct working of AIBO watchdog. The reasoning process during the experiment has performed the results as expected. The results conform the output of the designed reasoning system. Therefore, the fifth requirement:

- ➢ ***Requirement 5:*** Test this prototype to see whether the designed system and approach perform as expected.

has also been achieved. The prototype was tested by four test scenarios. These scenarios require reasoning of events in parallel or sequential, prioritizing scenarios and avoiding new introduced obstacles. Normally reasoning about events occurs in a sequential way, but to deal with parallel inputs two memory components were introduced, the short term and long term memory. The most important scenario is placed in the short term memory and the other scenarios are placed in the long term memory. The physical reactions are not always executed properly. The physical turning and walking of AIBO watchdog has some deviations and the amount of deviation is dependent on the floor where AIBO watchdog is operating.

9.2. Recommendations

The implemented architecture contains the minimal functionality required to show adequate behaviors of the AIBO watchdog in a real environment. Because of the large size of the project some simplifications have been made in some of the implemented architectures components. Therefore there is a lot of space for improvements. In this section first recommendations are discussed for the components of the designed architecture that have not been implemented and afterwards recommendations for the implemented components are presented.

9.2.1. Recommendations for not Implemented Components

There are two components in the developed architecture that are not implemented, internal needs and actions attributes XML.

Internal needs

The internal needs component deals with the emotion and needs of the AIBO watchdog. When the battery level of AIBO is low AIBO should react on this internal event appropriately. This whole concept can be developed with help of the work of Iulia Dobai, Personality model for a companion AIBO [49]. In this work many concepts have been proposed and their correctness has been proved.

Actions attributes XML

The actions attributes XML component provides the possibility for the user to change the actions attributes of the AIBO watchdog. These actions can be walking a certain amount of distance, barking with a certain sound wave or storing the captured picture in a certain directory. By storing these attributes in an XML file the user can adjust these values very easily to their desired settings.

9.2.2. Recommendations for Future work

Improvements in the perceiving component

In the current implementation, the AIBO watchdog receives the features information of objects in its environment from an external PC every time when AIBO reaches a waypoint. A better method is using the available camera to capture the image and extract these features automatically. The available microphones should capture the sound, analyze it and estimate the sound direction. As an intermediate step one can consider to implement a program which is able to recognize icons instead of real objects. The iconic representation of an object can be attached to a box. This box represents the object that the icon describes. This approach is much easier to realize, since real objects can be very big or very small and for iconic representation their size can be the same and many other object characteristics do not need to be taken into account, such as shadows.

Improvements in the interpretation component

The number of perceived features of objects must be sufficient to recognize the original objects. The improvements of this component are heavily dependent on the perceiving component. For the best result it is advised to develop these 2 components together. If AIBO is only able to distinguish 10 colors and 4 shapes, then AIBO can classify 40 objects at most. Therefore developing new methods to perceive new features is necessary. When the feature space becomes too large, one can consider using a neural network to classify the objects instead of an expert system.

Improvements in the scenario reasoning component

In the design it was chosen for the dynamic scripting approach to determine the current scenario. Based on the incoming objects the chances of the scenarios is adapted. The scenario which has reached the threshold and has a higher chance score is chosen. This approach can be replaced by a probabilistic approach, the Bayesian Network. By its probability model it is one of the best models to operate in an uncertain environment. In an uncertain environment nothing can be concluded for sure, but it is possible to conclude that this scenario fits 80% of the inputs. Based on the inputs a better understanding of the environment is achieved by calculating more accurate probability of all scenarios, but it is not easy to determine the probability values between a certain object and scenarios. A lot of research needs to be completed to be able to determine the probability values.

Improvements in the reaction reasoning component

After understanding the current environmental situation AIBO needs to know how to react on it. The chosen approach uses an expert system to determine the appropriate reactions. The reaction reasoning component can be merged with the scenario reasoning component to determine the reactions. A Bayesian Network can replace both components and produce the actions immediately. The number of reactions should be extended to create more natural reactions of the AIBO watchdog.

Improvements in world navigation

During the test of the AIBO watchdog the navigation algorithm has produced the correct outputs, but there are small deviations of the physical movements of AIBO. When the navigation algorithm wants to turn 90 degrees to the left, AIBO will make a turn of 85 degrees which results a deviation of 5 degrees to the right. To solve this problem AIBO should calibrate itself when it discovers that it is not moving correctly. An algorithm to let AIBO localize itself based on the features in its environment is the ideal situation. This approach requires accurate sensors inputs. Good lighting is the basic requirements for this approach.

The used navigation algorithm takes only four walking direction into account at each waypoint: walk forward, walk backward, turn left and turn right. Besides this algorithm a more accurate algorithm is required to navigate more efficiently in its environment without bumping into objects. This approach requires accurate physical movements of the AIBO. It is required that AIBO can turn at least multiple of 15 degrees, then diagonal movements can also be executed. Since the current algorithm turns the AIBO multiple of 40 degrees it is very hard to let AIBO navigate correctly to its destination. The walk functionality also needs to be improved. AIBO needs to be able to walk straight at all kind of floors. Furthermore AIBO watchdog should be able to walk appropriately on sloping roads and bumpy ones. Some researches about walking on sloping and bumpy roads for an AIBO robot dog has already been conducted, but more research is still required [9].

Improvement in action execution

The implemented action executor executes actions only one by one. In future it must be possible to execute actions in parallel. Two or more consecutive actions can be executed in parallel, if they are independent of each other and the place of execution is not an important issue. For example, creating an alarm sound and walking to the door can be executed at the same time.

Autonomous AIBO watchdog

When the improvements have already been made, the ultimate goal of the AIBO watchdog project is to let AIBO patrol the environment on its own without help of other tools. An external PC to control the AIBO watchdog should be redundant. In order to achieve that all programs should be able to run on the Apertos operating system of AIBO, fit on the 16MB memory stick and run on its 64MB working memory.

The designed architecture provides lots of opportunities for improvements, such as a better navigation method in the environment, an improved object recognition functionality and a more sophisticated reasoning system. The implemented architecture is only a proof of concept. It is required to do real life testing instead of simulation to gather more requirements based on real life situations. Technology is developing in a rapid pace; with better specification of the robots in future it is undoubtedly possible to realize such watchdogs. This project has indicated the limitations of the current technology, but utilizing the current technology we have brought the autonomous robot watchdogs' concept a step closer.

Appendix A

Jess – A Rule-Based System

In this appendix the characteristics of Jess will be discussed. Jess stands for Java Expert System Shell and it is a rule engine and scripting language developed at Sandia National Laboratories in Livermore, California in the late 1990s. Jess is written in Java, so it is an ideal tool for adding rules technology to Java-based software systems.

Jess is dynamic and Java centric, so it automatically gives you access to all of Java's powerful APIs for networking, graphics, database access, and so on. Jess has been used to develop a broad range of commercial software, including:

- Expert systems that evaluate insurance claims and mortgage applications.
- Agents that predict stock prices and buy and sell securities.
- Design assistants that help mechanical engineers.
- Smart network switches for telecommunications.

Jess can be used in command-line applications, GUI applications, servlets, and applets. Jess is a very flexible system. Its applications can be developed without compiling a single line of Java code or they are controlled entirely by Java code. Jess has been deployed in everything from enterprise applications using J2EE on mainframes to personal productivity applications on handheld devices.

Jess's rule engine uses an improved form of a well-known method called the Rete algorithm (Rete is Latin for net) to match rules against the working memory. The Rete algorithm explicitly trades space for speed, so Jess can use a lot of memory. Jess does contain commands that let you sacrifice some performance to decrease memory usage. The power of the Rete algorithm is that it 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. Furthermore, Rete organizes the pattern matcher so that these few facts are only tested against the subset of the rules that may actually match.

Appendix B

In this appendix the pictures will be presented that are made during the test scenarios. During the test scenarios, which have been mentioned in chapter 7, some pictures have been made by the AIBO watchdog. These pictures give an impression of the AIBO watchdog in practice. Figure B.1 shows the pictures of the detected fire by the AIBO watchdog.

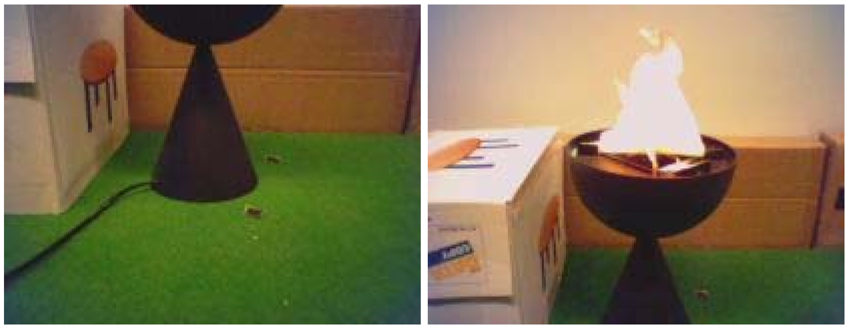


Figure B.1: Pictures of detected fire taken by AIBO. The left picture shows a picture of the fire object made by AIBO during patrolling. The right picture shows the fire object itself.

Figure B.1 illustrates the problems to discover the full object with a camera. A focusing algorithm should be developed to focus on the full object when AIBO takes a picture. Figure B.2 illustrates the same problem but with a different object.

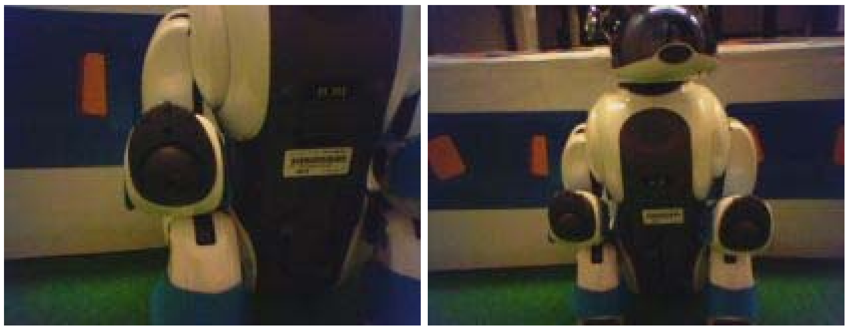


Figure B.2: Pictures of detected owner taken by AIBO. The left picture shows a picture of the owner object made by AIBO during patrolling. The right picture shows the owner object itself.

These two pictures show a problem when AIBO is taking a picture of a certain object. It is insufficient to take a picture only, it is also necessary to find the shape of the object and focus on that object. These two actions should be implemented when the object simulation approach by the program are replaced by the real object recognition approach. People also have to take into consideration that multiple image objects can be detected in one image. Therefore it is necessary to understand that the shape finding algorithm can find multiple shapes, but some shapes are objects that are very far away. This issue needs to be solved when we want to work with real object recognition.

Appendix C

In this appendix the implemented reactions of the AIBO watchdog will be explained. These reactions can be placed in the implemented expert systems that find the corresponding reactions for every scenario. Table C.1 shows the implemented reactions and a short description is provided.

These actions can be placed in the actions reasoning expert system, evenhandler.clp file, to define the desired reactions of people for the AIBO watchdog.
Figure C.1 shows a rule in the eventhandler.clp file.

```
(defrule fire
(actionadder(mainevent "FIRE")(done "FALSE")(OBJECT ?w))
=>
(?w setDone "TRUE")
(?w setStack "turntoobject")
(?w setStack "headscan")
(?w setStack "capturescene")
(?w setStack "email")
(?w setStack "alarm")
(?w setStack "runtocenter")
)
```

Figure C.1 A rule in eventhandler.clp that will place reactions on the stack if the fire scenario is active.

By replacing the string behind the setStack command a new action will replace the old one. To add more reactions for a certain situation, one has to add more setStack lines in the rule. Removing the setStack line will reduce the number of reactions.

Table C.1: Implemented reactions for the expert system.

Low level actions	
walk	Each paw of AIBO will move 3 steps forwards from its origin.
headscan	The head of AIBO will move to left and right and it stops in the neutral zone.
turnleft	Each paw of AIBO will move 2 steps and AIBO will turn 80 degrees to the left.
turnright	Two paws of AIBO will move 2 steps and the other two paws will move 3 steps. AIBO will turn 85 degrees to the right.
turn180	Each paw of AIBO will move 5 steps and AIBO will turn 190 degrees.
capturescene	A picture will be taken and saved in the main directory of the program.
capturesound	A picture will be taken and saved in the main directory of the program. *NOT IMPLEMENTED YET*
alarm	AIBO will create a barking sound.
alarm2	AIBO will create a big alarming sound.
testdistance	AIBO will measure the distance to its nearest object. If the distance is smaller than 30cm, an unknown object will be placed in the environment map and a new direction will be calculated.
email	An email will be sent to a certain person. *NOT IMPLEMENTED YET*
High level actions	
turntonode	The next node on the path will be calculated and AIBO will turn to the direction of the path.
turntoobject	AIBO will turn itself to the detected object direction.
turntosoundobject	AIBO will turn itself to the detected sound direction.
runfromfire	The new goal point is the default goal node (3,5), if there are no escape objects were encountered. Otherwise the goal point will be the location where AIBO discovered the escape object.
runfromintruder	The new goal point is the default goal node (3,1).
runfromringingbell	The new goal point is the default goal node (7,7), if there are no notify objects were encountered. Otherwise the goal point will be the location where AIBO discovered the notify object.
runfrombrokenglass	The new goal point is the default goal node (4,1).

Reference

[1]. Violent offences top million mark: http://news.bbc.co.uk/2/hi/uk_news/politics/4700575.stm

[2]. FBI Releases its 2005 Crime Statistics: http://www.fbi.gov/ucr/05cius/about/crime_summary.html

[3]. U.S. and World Population Clocks – POPClocks: http://www.census.gov/main/www/popclock.html

[4]. Toename diefstal jonge auto's: http://www.rtl.nl/(/actueel/rtlboulevard/crime/)/components/actueel/rtlboulevard/2006/07_juli/crime/060714_auto.xml

[5]. Man dringt residentie Britse koningin binnen: http://www.nu.nl/news.jsp?n=415398&c=21

[6]. The incredible human body: The five senses: http://school.discovery.com/lessonplans/programs/humanbody/

[7]. Official Sony AIBO website: http://support.sony-europe.com/aibo/index.asp

[8]. Joanne Pransky, "AIBO- the Nr. 1 selling service robot.", Industrial Robot: an international journal, Volume 28, Number 1 2001, pp24-26.

[9]. Yuki Naoda, Daiki Satoh, Miyuki Fujii and Michiko Matsuda, "Automatic Walking-Pattern Acquisition on a punishing road for a quadruped robot: AIBO."

[10]. Laurence Nigay, Joelle Coutaz, "A design space for multimodal systems: concurrent processing and data fusion", Interchi "93, 24-29 April 1993.

[11]. Assuralia, "Verzekering en preventie bij woningen.": http://www.assuralia.be/nl/branches/batibouw/liste.asp

[12]. Home safety and home security: http://www.safewithin.com/homesafe/home.sec.cgi?1,1

[13]. Vakantie beurs. "Je huis veilig achterlaten." : http://www.vakantiebeurs.nl/reisinformatie/reisinformatie_algemeen.php?cat=huisachterlaten

[14]. Wereldreisgids. "Uw huis achterlaten.": http://www.wereldreisgids.nl/content/new-main.asp?id=82

[15]. Gemeente Utrecht. "Utrecht steeds veiliger.": http://www.utrecht.nl/smartsite.dws?id=131024

[16]. Fatiha Mamache, "Representing temporal relationships between events and actions."

[17]. Doreen Massey, "Space-time, 'Science' and the relationship between physical geography and human geography", June 1999.

[18]. Peter Shirley. "Fundamentals of computer graphics, 2nd Edition." A.K. Peters Ltd.

[19]. Stanford encyclopedia of philosophy. “Memory”: http://plato.stanford.edu/entries/memory/

[20]. ” Mnemosyne, Human ecology of memory”: http://socrates.berkeley.edu/~kihlstrm/mnemosyne.htm

[21]. Jeff Hawkins with Sandra Blakeslee, “On intelligence”, New York: Times Books.

[22]. Byoung-Ju Lee and Gwi-Tae Park, “A Robot in Intelligent Environment: Soccer Robot”, Advanced Intelligent Mechatronics, 1999. Proceedings. 1999 IEEE/ASME International Conference, 1999, pages 73-78.

[23]. Kikuo Fujimura and Hanan Samet, “A hierarchical strategy for path planning among moving objects”, Robotics and Automation, IEEE Transactions, Feb 1989, pages 61-69.

[24]. Larman, C., “Applying UML and Patterns”.

[25]. Creating Use Case Diagrams: http://www.developer.com/design/article.php/10925_2109801_1

[26]. Practical UML: A Hands-On Introduction for Developers: http://bdn.borland.com/article/0,1410,31863,00.html

[27]. Michael Luck, Ronald Ashri, Mark D’Inverno, “Agent-based software development”, Artech House Publishers, 2004.

[28]. Wooldridge, M., and N.R. Jennings, “Agent Theories, Architectures, and Languages: A survey”, Intelligent Agents: Theories, Architectures, and Languages, M. Wooldridge and N. Jennings, volume 890 of LNCS, New York: Springer, 1995, pp. 1-39.

[29]. Brooks, R. A., “A robust Layered Control System for a Mobile Robot,” IEEE Journal of Robotics and Automation, Vol. 2, No. 1, 1986, pp14-23.

[30]. Ferguson, I. A., “TouringMachines: an architecture for dynamic, Rational, Mobile Agents,” Ph.D. thesis, Clare Hall, University of Cambridge, England, 1992.

[31]. Kiss, G., “Goal, Values, and Agent Dynamics,” Foundations of Distributed Artificial Intelligence, G. O’Hare and N. Jennings, New York: John Wiley and Sons, 1996, pp. 247-268.

[32]. Cohen, P.R., and H. J. Levesque, “Intention is choice with commitment,” Artificial Intelligence, Vol. 42, 1990, pp. 213-261.

[33]. Sabrina Sestito, Tharam S. Dillon, “Automated Knowledge Acquisition”, Prentice Hall of Australia Pty Ltd, 1994.

[34]. Han Reichgelt, “Knowledge Representation: an AI perspective”, Ablex Publishing Corporation, 1991.

[35]. Hector J. Levesque and Gerhard Lakemeyer, “The logic of knowledge bases”, The MIT Press, 2000.

[36]. Peter Jackson, “Introduction to expert systems.”, 2nd edition, Addison-Wesley Publishing company, 1990.

[37]. Sankar K. Pal, Tharam S. Dillon, and Daniel S. Yeung, “Soft computing in Case Based Reasoning.”, Springer-Verlag London Limited, 2001, Julie Main, Tharam S. Dillon, and Simon C.K. Shiu, “A tutorial on Case Based Reasoning”.

[38]. Stuart Russell, Peter Norvig, “Artificial Intelligence: a modern approach”, 2nd edition, Pearson Education Inc, 2003.

[39]. Gene Bellinger, Durval Castro, Anthony Mills, “Data, Information, Knowledge, and Wisdom”: http://www.systems-thinking.org/dikw/dikw.htm.

[40]. Silvia Oana Tanase, “Bachelor Thesis: AIBO WatchDog”, EEMCS, TU Delft, 2005.

[41]. Brian Smith, “Reflection and semantics in a procedural language. Ph. D. thesis and technical report MIT/LCS/TR-272, MIT, Cambridge, 1982.

[42]. Fikes, R., & Kehler, T., “The role of frame-based representation in reasoning.”, Communications of the ACM, 28, 904-20.

[43]. Lindsay, R., “Inferential memory as the basis of machines which understand natural language.”, E. Feigenbaum & J.Feldman (Eds.), Computers and thought, New York: McGraw-Hill.

[44]. Aleksander, I. 1989a, “Connectionist systems: information technology goes brain-like (again!)”, in Intelligent systems in a human context: Development, implications, and applications, eds L.A. Murray & J.T.E. Richardson, Oxford University press, Oxford, pp. 47-52.

[45]. Wasserman, P.D. & Schwartz, T. 1988, „Neural networks, part2“, IEEE expert, Spring, pp10-15.

[46]. Touretzky, D.S. & Pomerleau, D.A. 1989, „What’s hidden in the hidden layers?”, Byte, August, pp. 227-33

[47]. RoboCup competition: http://www.tzi.de/4legged/bin/view/Website/WebHome

[48]. RoboCup.nl: http://www.robocup.nl/

[49]. Iulia Dobai, “Personality model for a companion AIBO.”, EEMCS, TU Delft, 2005.

[50]. Francisco Martin Rico, Rafaela Gonz_alez-Careaga, Jose Maria Canas Plaza,Vicente Matellan Olivera, “Programming Model Based on Concurrent Objects for the AIBO Robot”, Actas de las XII Jornadas de Concurrencia y Sistemas Distribuídos, Universidad Rey Juan Carlos y Universidad Complutense de Madrid, junio 2004.

[51]. Iulia, Zhenke, ”OPEN-R development tools overview”, EEMCS, TU Delft, 2005.

[52]. Ferguson, I.A., “Integrated control and coordinated behavior: a case for agent models,” Intelligent agents: Theories, architectures, and languages, M. Wooldridge and N. Jennings, (eds), Volume 890 of LNCS, New York: Springer, 1995, pp. 203-218.

[53]. Harvey Fletcher, “Auditory patterns”, Bell telephone laboratories, New York.

[54]. Carlos Martinho, Ana Paiva, “Pathematic Agents: Rapid Development of Believable Emotional Agents in Intelligent Virtual Environments”, Technical University of Lisbon and Instituto de Engenharia de Sistemas e Computadores.

[55]. Marvin Minsky, "A Framework for Representing Knowledge", in Patrick Henry Winston (ed.), The Psychology of Computer Vision, McGraw-Hill, New York, 1975.

[56]. Bayu Slamet, Arnoud Visser, “Purposeful perception by attention-steered robots”.

[57]. Raffay Hamid, Aaron Bobick, Anthony Yezzi, “AUDIO-VISUAL FLOW - A VARIATIONAL APPROACH TO MULTI-MODAL FLOW ESTIMATION”, Image Processing, 2004. ICIP '04. 2004 International Conference on Volume 4, 24-27 Oct, 2004 Page(s):2563 - 2566 Vol. 4.

[58]. Gun A. Lee, “Improving AIBO with Artificial Intelligence Technique”, Term paper on A.I. course, Dec.12, 2002.

[59]. Kohtaro Sabe, Masaki Fukuchi, Jens-Steffen Gutmann, Takeshi Ohashi, Kenta Kawamoto, and Takayuki Yoshigahara. “Obstacle Avoidance and Path Planning for Humanoid Robots using Stereo Vision”, Robotics and Automation, 2004, Proceedings. ICRA '04. 2004 IEEE International Conference on Publication Date: 26 April-1 May 2004, Volume: 1, On page(s): 592- 597 Vol.1.

[60]. Bram Bakker, Viktor Zhumatiy1, Gabriel Gruener3, Jürgen Schmidhuber, “A Robot that Reinforcement-Learns to Identify and Memorize Important Previous Observations”, In "IEEE/RSJ International Conference on Intelligent Robots and Systems", IEEE, 2003.

[61]. Richard Washington, Keith Golden, John Bresina, David E. Smith, Corin Anderson, Trey Smith. “Autonomous Rovers for Mars Exploration”, Proceedings of the IEEE Aerospace Conference, 1999, IEEE, 1999.

[62]. Ingeborg Strand Friisk, “Autonomous AIBO watchman”, Norwegian University of Technology and Science, NTNU, 2003.

[63]. Ioana Butoi, “Find Kick Play An Innate Behavior for the Aibo Robot”, Senior Thesis, spring 2005.

[64]. Mickaël THOUZERY, “Fusing speech and face recognition on the AIBO ERS-7”, EEMCS, TU Delft, 2005.

[65]. Harm Aarts Jelmer de Vries Jan-Willem van den Broek, “Aibo Programming Report of Group Vision1b”, 2004.