Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

An Implementation Framework for Emotion Based Adaptive Agents

Pedro Brilhante das Neves

Mestrado em Engenharia Informática

2008

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

An Implementation Framework for Emotion Based Adaptive Agents

Pedro Brilhante das Neves

PROJECT

Projecto orientado pela Prof[a]. Dr[a]. Maria da Graça Figueiredo Rodrigues Gaspar
e co-orientado pelo Prof. Dr. Luís Filipe Graça Morgado

Mestrado em Engenharia Informática

2008

Declaração

Pedro Brilhante das Neves, aluno nº 27836 da Faculdade de Ciências da Universidade de Lisboa, declara ceder os seus direitos de cópia sobre o seu Relatório de Projecto em Engenharia Informática, intitulado "An Implementation Framework for Emotion Based Adaptive Agents" , realizado no ano lectivo de 2007/2008 à Faculdade de Ciências da Universidade de Lisboa para o efeito de arquivo e consulta nas suas bibliotecas e publicação do mesmo em formato electrónico na Internet.

FCUL, September 30, 2008

Profª. Drª. Maria da Graça Figueiredo Rodrigues Gaspar, supervisora do projecto de *Pedro Brilhante das Neves*, aluno da Faculdade de Ciências da Universidade de Lisboa, declara concordar com a divulgação do Relatório do Projecto em Engenharia Informática, intitulado "An Implementation Framework for Emotion Based Adaptive Agents".

Lisbon, September 30, 2008

Resumo

Está a ser desenvolvido, na unidade de investigação LabMAg, o projecto "AutoFocus: Adaptive Self-Improving Multi-Agent Systems", no qual o presente trabalho de mestrado se enquadra. O projecto AutoFocus tem como objectivo a implementação de sistemas multi-agente baseados em entidades autonómicas capazes de comportamentos auto-optimizados e adaptativos.

A noção de computação autonómica, tal como outras noções que também implicam computação pró-activa, baseia-se em entidades autónomas que agem activamente no sentido de alcançar os seus objectivos e que têm a capacidade de se adaptar dinamicamente a mudanças no seu ambiente, restringidas por limites de tempo e de recursos. Na abordagem do projecto AutoFocus essa adaptação à mudança, assim como a regulação das capacidades dos agentes, é resultante da combinação de aspectos cognitivos com aspectos de base emocional. O modelo de agente subjacente ao projecto AutoFocus é o Modelo de Agente de Fluxo.

A tarefa a que correspondeu este projecto de mestrado, consistiu em desenvolver uma plataforma de implementação para o Modelo de Agente de Fluxo. Pretendeu-se com esta plataforma disponibilizar uma ferramenta que permita a rápida implementação de agentes baseados neste modelo bem como a sua monitorização.

O trabalho desenvolvido consistiu na análise e desenho, orientado a objectos, implementação e teste dos componentes desta plataforma.

PALAVRAS-CHAVE:
inteligência artificial, agentes inteligentes, modelo cognitivo, modelo de emoção, plataforma de experimentação

Abstract

The work presented in this document is part of the project "AutoFocus: Adaptive Self-Improving Multi-Agent Systems" that is being developed at the research unit LabMAg, which objective is the implementation of multi-agent systems based on autonomous entities capable of self-optimized and adaptive behaviors.

The notion of autonomic computation, like other notions that also imply proactive computation, is based on autonomous entities that actively work to achieve their objectives and have the ability to dynamically adjust to changes in their environment, constrained by time and resource limits. In the approach used by the AutoFocus project, that adaptation to change and the regulation of the agent's capabilities, result from the combination of cognitive aspects with emotional based aspects. The agent model defined and used by the AutoFocus project is the Agent Flow Model.

The task that corresponded to the work presented in this document was to develop a platform for the Agent Flow Model. It was intended, with this platform, to provide a tool that enables the rapid deployment and monitoring of agents based on this model.

The developed work consisted in the analysis and design, oriented to objects, implementation and testing of components of this platform.

KEYWORDS:
artificial intelligence, intelligent agents, cognitive model, emotion model, experimentation platform

Contents

List of Figures

Chapter 1

Introduction

This initial chapter presents the objectives of this project in the context of the AutoFocus project and briefly describes the organization of this document.

1.1 Context

The AutoFocus project main goal is to develop an agent model and architecture capable of:

(i) creating the necessary support for real time adaptation an learning, according to the agent's experience;

(ii) regulating the agent's internal processes, according to its resources and time constraints.

For these purposes the Agent Flow Model was developed by Prof. Luís Morgado and introduced in his PhD thesis[6] and several published articles (e.g.[7][8]), co-authored by Prof. Graça Gaspar.

In the Agent Flow Model the regulation of the agent's internal processes is achieved through emotion based mechanisms. These mechanisms regulate the amount of time and resources used by the agent's cognitive processes and the formation of internal memories.

1.2 Objectives

Although several Agent Flow Model prototypes already exist, the key features that compose this model, namely the cognitive structure and the base mechanisms, have been specifically implemented for each prototype according to the problem addressed, thus not representing a general solution.

This project was idealized to integrate the knowledge gathered from those prototypes and to provide a general reusable implementation of the key features of the

Agent Flow Model. More specifically, to build a computational library to serve as a tool for the rapid deployment and monitoring of agents based in this model.

1.3 Document Organization

This document is organized as follows:

- Chapter 2 briefly describes the Agent Flow Model supporting theories: the conceptual spaces and the emotion model used.
- Chapter 3 introduces the Agent Flow Model architecture general view and the decomposition of this model in three main constituents: the cognitive structure, the base mechanisms and the cognitive processes.
- Chapter 4 describes the work done in designing and implementing the AutoFocus platform:
  - Section 4.1 presents the project objectives in more detail and the planning and methodology used;
  - Section 4.2 describes the general platform conception and how it was divided into three subsystems: Agent, Environment and AutoFocusGlue;
  - Section 4.3 presents the AutoFocus domain model;
  - Section 4.4 explains, in detail, the interactions between the Agent, Environment and AutoFocusGlue systems;
  - Section 4.5 introduces the agent class model and the base mechanisms algorithm;
  - Section 4.6 presents the Experiment Program as a general tool to control the AutoFocus platform execution;
  - Section 4.7 presents the prototype implemented to test the platform, the Tileworld agent, and the results obtained.
- Chapter 5 presents the conclusion and the future work related to the AutoFocus platform.

Chapter 2

Supporting Theories

This chapter briefly introduces the main notions upon which the agent flow model and architecture was defined.

2.1 Conceptual Spaces

The cognitive sciences have two objectives: the explanation of the cognitive activity through theories and the construction of artifacts that can accomplish those activities. Artificial Intelligence focuses mostly in the last one and for that purpose, there are two main approaches to represent cognition from a computational point of view, the symbolic approach and the associationist approach.

The symbolic approach consists, essentially, in symbol manipulation according to explicit rules, while the associationist approach focuses on the associations among different kinds of information elements to represent cognition. Though both approaches have their advantages and disadvantages, neither can perform reasonably well the task of concept learning, which is closely tied to the notion of similarity, central to a large number of cognitive processes.

To overcome these difficulties Gärdenfors[4] purposes another form of representation, the conceptual representation, based on geometrical structures, where similarity relations can be modeled in a natural way.

Quality Dimension

The key notion of this new representation is that of quality dimension, whose role is to build up the domains needed for representing concepts. Some examples of quality dimensions are temperature, weight or the three ordinary spatial dimensions height, width and depth. The main function of these dimensions is to represent various "qualities" of objects. For example, one can judge tones by their pitch, for which our perception recognizes an ordering from "low" to "high" tones.

The dimensions form the framework used to assign properties to objects and to

specify relations among them. The coordinates of a point within a conceptual space represent particular values on each dimension, for example, a particular temperature, a particular height, and so forth. It is assumed that each of the quality dimensions is equipped with certain geometrical structures, like an ordering or a metric. For example the dimension *weight*, illustrated in figure 2.1[4], is a positive continuous ordered dimension.

Figure 2.1: The weight dimension.

Certain quality dimensions are integral in the sense that one cannot assign an object a value on one dimension without giving it a value on the other. Dimensions that are not integral are considered separable.

A domain is a set of integral dimensions that are separable from all other dimensions. The main reason for decomposing a cognitive structure into domains is the assumption that an object can be assigned certain properties independently of other properties.

Conceptual Space

A conceptual space is defined as a collection of one or more domains. A point in space will represent an object or concept depending on the context in which they are used. While an object refers to a particular artifact, a concept is an idea that characterizes a set, or category, of objects.

It is possible to take a particular perspective of a concept by giving some domains particular attention. This is accomplished by assigning different weights to different domains.

A property is defined with the aid of a single dimension or domain. The main idea is that a property corresponds to a region (subspace) of the conceptual space. In contrast, a concept may be based on several separable subspaces. Properties form a special case of concepts.

Conceptual spaces are static in the sense that they only describe the structure of representations. This notion of conceptual spaces, as defined by Gärdenfors[4], served as inspiration for the definition of the conceptual structure of the Agent Flow Model, defined by Luís Morgado in his PhD thesis[6], that is the background for the work presented here.

2.2 The Emotion Model

The subjective nature of emotions makes them difficult to characterize, so an explanation is in order of what exactly are we talking about and of the context in which the term emotion is used in this work.

In the following, I will not attempt to present the different perspectives of emotion that exist today but rather I only intend to introduce the ideas behind the emotion model upon which the Agent Flow Model was defined.

Cognition

From a classic perspective, emotion requires a minimum level of cognition, which presupposes a brain structure that only some living beings, like humans and other mammals, have. However, if we consider the perspective defended by Maturana and Varela[5], cognition can be defined as the "effective action of a living being in its environment". This means that cognition is a common property shared by all living organisms and can be seen in the organisms capacity to execute actions that allow them to strive, by adapting to their environments ever changing conditions. In this perspective, simple organisms, like a bacteria or a plant, are capable of cognition and action.

Biologic Systems and Autopoiese

One of the main characteristics of the living beings is their capacity to continually recreate themselves. For example, in complex organisms, tissues and organs substitute their own cells in continual cycles, maintaining, at the same time, their integrity as a whole. This capacity of dynamic self-creation is designated as autopoiese by Maturana e Varela[5].

In a autopoietic system each component participates in the creation or transformation of other components of the system, in a network of interdependence, allowing the system to continuously create itself.

This process begins with the differentiation of the body in relation to the surrounding environment through a dividing structure, such as the cell membrane. It's this membrane that allows the internal organization of the body, which in turn generates it. So, we are not dealing with two separated processes, but rather two distinct aspects of the same phenomenon. The interruption of any of the processes would lead to the end the organism[5].

Therefore there is a cyclical relationship of feedback in autopoetic systems, where each component affects the other which, in turn, affects the former. A central feature of the feedback cycles is the ability to self-regulate, either by maintaining a stable internal environment, or by the generation of actions of the system, allowing the

continuous viability of the global body.

Auto-regulation and Motivation

With the differentiation between the interior and exterior of the body, through the mechanisms of self-regulation, all variables that define the inner state are independent of the ones that define the exterior. This means that autopoetic systems are autonomous in nature, which translates to pro-active behaviors, motivated by their self-regulating processes. These behaviors arise from the need to control and maintain the organism integrity.

To this end, the mechanisms of self-regulation regularly monitor the environment, comparing the values observed with benchmarks, triggering the necessary steps to reduce the difference observed. This difference represents the motivation of the organism.

The thermodynamic paradox

The fact that living beings are able to create and maintain an organized structure, away from equilibrium with the environment, is in apparent opposition to the second law of thermodynamics, according to which, in a closed system, the entropy can only increase. This means that the nature tends to homogenization, i.e. change occurs naturally from order to chaos. Instead, living beings have the ability to create order from chaos, which goes precisely in the opposite direction.

One solution to this problem introduces the concept of dissipative structure. This structure would be an open system through which energy and matter flows along and where the internally generated entropy would be sent out of the system to ensure its continuity. For example, plants and animals absorb energy and matter of low entropy, in the form of light or food, and export matter of high entropy in the form of waste.

Agent as a dissipative structure

The notion of dissipative structure was chosen by Luís Morgado[6] as the appropriate support for modeling an agent that incorporates all three basic characteristic of biological systems described above: autopoiesis, self-regulation and motivation.

Based on the concept of dissipative structure, an agent is characterized by a set of internal potentials $\{p_1, p_2, ..., p_m\}$ and a set of flows $\{f_1, f_2, ..., f_m\}$, as depicted in figure 2.2 taken from [6].

The internal potentials define the internal structure of the agent, varying according to the internal activity, which in turn is governed by the maintenance of a specific internal organization of those same potentials.

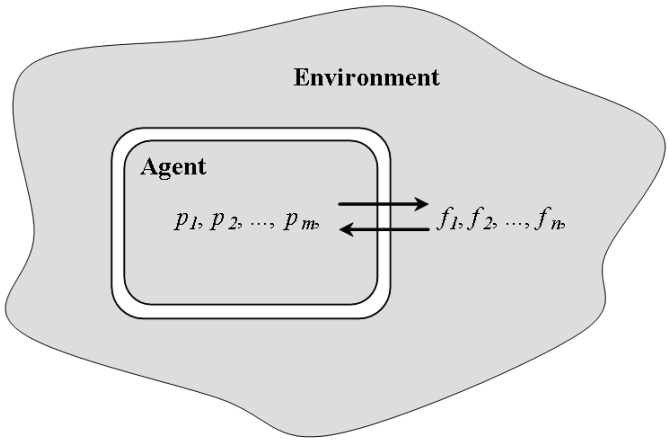

Figure 2.2: Agent as a dissipative structure.

Motivation formation

The maintenance of a viable structure, despite variations in the environment, means that the internal potentials are kept within viability limits.

It is the act of maintaining the internal potentials within these limits, by processes of self-regulation, which is considered the primary source of motivation of the agent. The viability limits may be implied by structural restrictions, or set explicitly, in the form of potential regulators, and therefore explicitly influence the agents motivation and behavior.

The motivations can be distinguished in built-in motivations, embedded into the agent during its design and implementation, and acquired motivations, resulting from the default motivations and the interaction of the agent with the environment, forming a hierarchy, with the built-in motivations at the base[6].

It is the satisfaction of those motivations that produces the forces that direct the activity of the agent, which in turn will lead to the emergence of new motivations, in a process of self-regulation typical of autopoietic systems.

Achieving Motivations

In order for the motivations to be fulfilled the agent must have the ability to produce the required change, either internally or externally. Inspired by the classic definition of thermodynamics, where energy is the ability to produce work, in the Agent Flow Model, the ability to produce change is seen as being expressed in the form of energy flows or being accumulated in the form of potential energy, which tends to produce such flows. This potential energy translates in the potential capacity of an agent to achieve its motivations.

The ability to produce change can be described, generally, by a potential P, designated achievement potential. In turn, when acting on the environment, the

agent may find more or less resistance to the change that it is trying to achieve. That resistance is called *the achievement conductance C*.

In a dissipative structure the achievement potential can be seen as a force and the achievement conductance as a transport property. Applying an achievement potential P on an achievement conductance C, results in a flow F, called achievement flow, illustrated in figure 2.3 taken from [6].

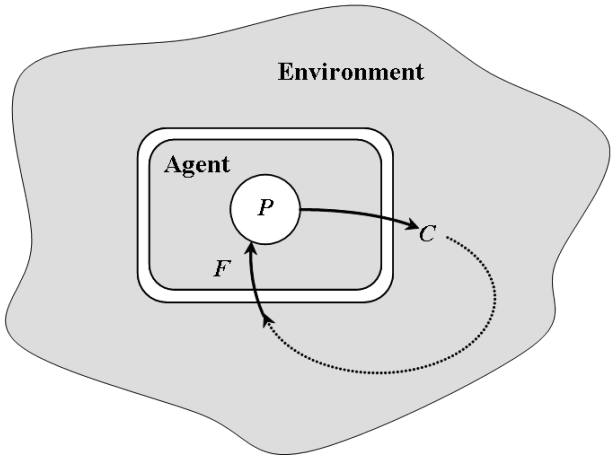

Figure 2.3: The relationship between agent and environment.

The Origin of Emotion

The achievement potential and flow represent, respectively, the motivational driving forces underlying the behavior of the agent and the relationship between agent and environment.

When the achievement potential is high, it means that the agent is capable of producing the change needed to achieve its motivations. On the other hand, if the achievement potential is low, the agent lacks that capacity.

The achievement flow expresses how the completion of the agent's motivations is evolving. If there is a favorable evolution of the completion of the agent's motivations, we will say the flow is *converging*, otherwise we will say the flow is *diverging*.

Looking at both the potentials and flows, we can identify four basic patterns of evolution of the agents situation:

- When the potential is high and the flow is convergent we have a favorable situation;
- When the potential is high and the flow is divergent we have a adverse situation;
- When the potential is low and the flow is divergent we have a situation of danger;

- When the potential is low and the flow convergent we have a situation of despondency.

These situations do not represent discrete states, but patterns of change involving both the dynamics of change and the agent's consequent behavior. This behavior is also determined by the nature of the agents, but should consist of some action in compliance with their motivations.

Making the bridge to the biological world, if we consider the four possible situations, we can identify in the living beings typical behaviors associated with each one of the situations.

- A favorable situation is associated with behaviors like approaching and enjoying;
- An adverse situation is associated with behaviors like mobilization and reaction;
- A situation of danger is associated with behaviors like self-protection and departure;
- A situation of despondency is associated with behaviors like inaction and recovery.

Comparing the four situations and types of behavior described above with the description featuring four basic emotions, by Ekman and Davidson[2], we get the correspondence visible on figure 2.4 taken from [6].

Achievement Potential	**Achievement Flux**	**Situation**	**Behavior**	**Emotion**
High	Convergent	Favorable	• Approach • Enjoy	*Joy*
High	Divergent	Adverse	• Mobilize • React	*Anger*
Low	Divergent	Danger	• Protect • Depart	*Fear*
Low	Convergent	Despondency	• Inaction • Recover	*Sadness*

Figure 2.4: The relationship between patterns of achievement potential and flow and basic emotions.

In this perspective, it is the whole formed by the expression of the motivational dynamics, actions and the subjective perception resulting therefrom, which may be characterized as emotion, illustrated in figure 2.5 taken from [6]. Thus, the emotional phenomena do not correspond to any type of mild cognitive representation. It's the dynamic evolution of the structure of an agent, and its relation with the environment that cause the perception of emotional patterns.

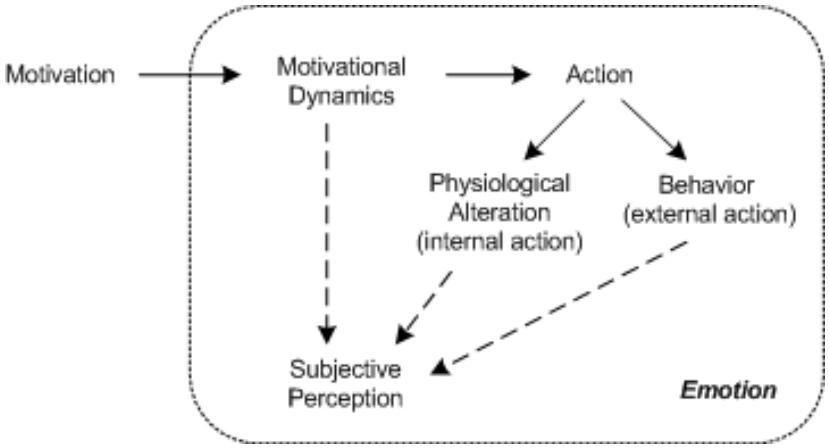

Figure 2.5: Emergence of emotion.

Emotional Dynamics

After discussing the origin of emotion we need to describe it in a concise and objective way to use it in a computational model.

The achievement potential and flow, from which emotion arises, vary in time, according to the agent behavior and its relation with the environment. These variations can be formally expressed by the achievement potential temporal variation (δP) and the achievement flow temporal variation (δF), respectively:

$$\delta P = \frac{dP}{dt} \quad and \quad \delta F = \frac{dF}{dt} \tag{2.1}$$

They are at the same time supplementary and mutually influential. This integrated dimensions are expressed through a vectorial function designated *emotional disposition* (ED).

$$ED \equiv (\delta P, \delta F) \tag{2.2}$$

While the *emotional disposition* function changes through time we can observe that for a specific time $t = \tau$, an *emotional disposition* vector is characterized by a quality, defined by the vector orientation and an intensity, defined by the vector size.

$$quality(ED) \equiv arg(ED) \tag{2.3}$$

$$intensity(ED) \equiv |ED| \tag{2.4}$$

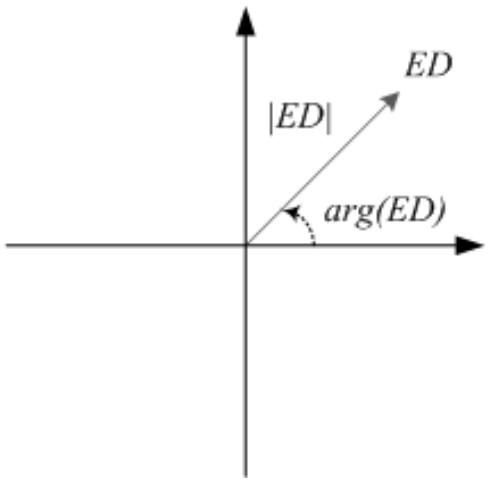

Figure 2.6: Emotional disposition vector.

So the notion of *emotional disposition* is composed by two distinct properties:

- *quality*: equivalent to the emotion character or pattern, as in figure 2.4 taken from [6].
- *intensity*: the emotion intensity or strength

From an emotional perspective it's possible to establish a correspondence between each quadrant of the two dimensional space $\delta P \times \delta F$ and the emotional patterns previously described. For example, in quadrant Q-I ($\delta P > 0$ and $\delta F > 0$) the achievement flow is convergent with the agents motivations and the positive achievement potential reflects a favorable evolution of the agents situation, what can be translated to the emotional pattern of *Joy* (see figure 2.7) taken from [6].

These emotional tendencies, associated to each quadrant, are only subjective indications of the essential nature of each quadrant since the quality of an emotional disposition is a continuous value.

It is important to note that the notion of emotional disposition does not constitute a direct analogy to the notion of emotion. Instead it's an action inducing mechanism in the same sense as a predisposition or readiness for action[3].

Figure 2.7: Relationship between the two dimensional space quadrants and the emotional disposition quality.

Chapter 3

The Agent Flow Model

The Agent Flow Model agent architecture, developed by Prof. Morgado[6], and depicted in figure 3.1, is composed by three main type of constituents: the cognitive structure, the base mechanisms and the cognitive processes (perception, assimilation, reasoning, decision and action).

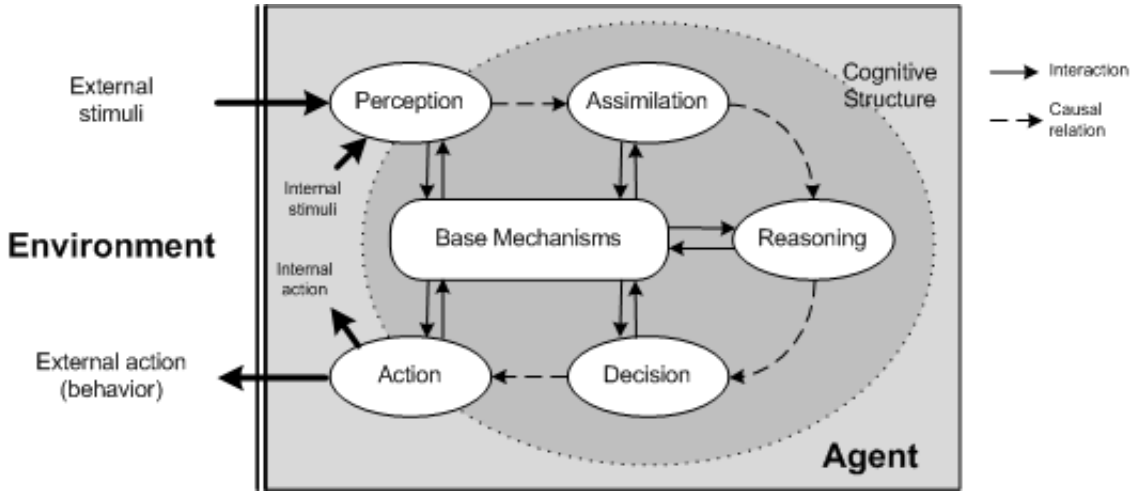

Figure 3.1: The Agent Flow Model architecture.

This is a general view of the architecture and it is not required that all aspects are present in every agent. For example, Reasoning and Decision processes will probably be absent in reactive agents. The same can be said for some of the base mechanisms. The main features of this agent model are the central usage of the cognitive structure to store and organize all the information maintained by the agent and how that information is used by the base mechanisms to obtain emotional dispositions and to regulate the cognitive processes.

3.1 The Cognitive Structure

Under the proposed model, the cognitive structure is composed by all the internal elements involved in the agents cognitive activity. These elements are modeled as a composition of internal potentials.

Cognitive Elements

The agent's potentials result from the interaction between the agent and the environment, and from the agents own internal activity. In any case, they express aspects, of the internal and external environment, that correspond to the quality dimensions (Gärdenfors[4]) that the agent is able to discriminate and understand. Since these potentials form the cognitive structure of the agent, namely in the form of memories, they are called cognitive potentials.

The cognitive potentials are a composition of two types of signals[6]:

(i) a qualitative signal $\varphi(t)$, that identifies the discriminated dimension;

(ii) a quantitative signal $\rho(t)$, corresponding to the value of the discriminated dimension.

At a certain time t, a cognitive potential p can be represented by:

$$p(t) = \rho(t)\varphi(t) \tag{3.1}$$

Through the aggregation of different cognitive potentials, differentiated by their corresponding dimension i, we get a cognitive element $\sigma(t)$, represented by:

$$\sigma(t) = \sum_{i=1}^{K} p_i(t) \tag{3.2}$$

where K is the number of aggregated cognitive potentials.

We can see in figure 3.2, taken from [6], an illustration of the formation of cognitive elements, in the perception context. We can identify three main activities involved on the agent perception:

- *detection*: where the outside signals, which may come in different forms, depending on the nature of the agent, are picked up by the agent;
- *discrimination*: in which sensory channels discriminate the different qualities of the signals, creating the respective cognitive potentials;

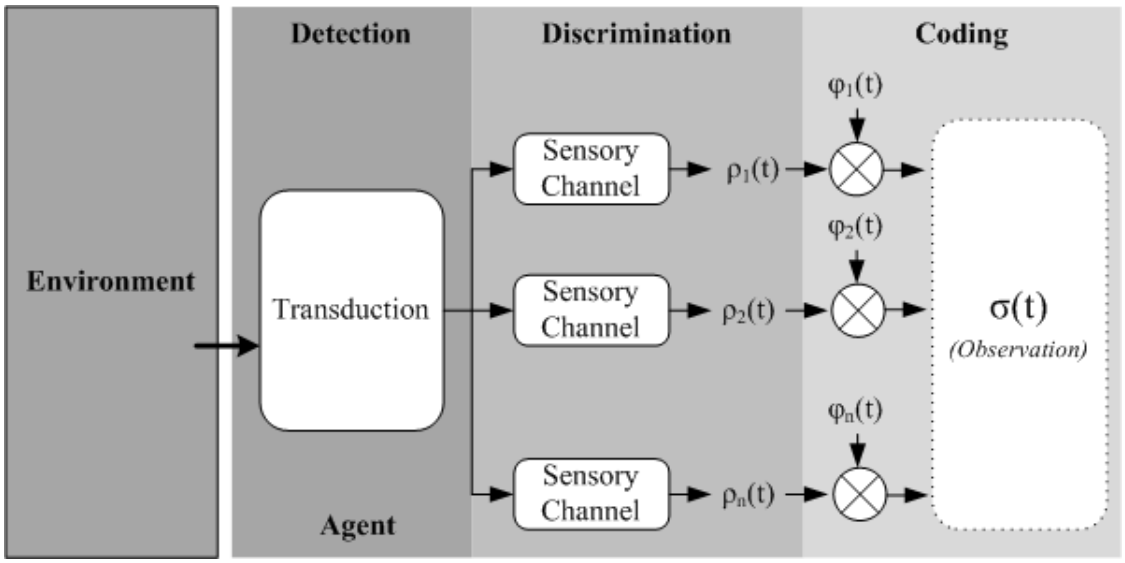

Figure 3.2: Formation of cognitive elements from the agent perception.

- *coding*: in which the cognitive elements are generated by manipulation and aggregation of the cognitive potentials previously created.

In this case the cognitive element, generated by the perception, is an observation and it has a very specific role in the agents model. But the same cognitive element can have a different role, depending on the context in which it is created or manipulated. The cognitive elements can play three main roles. They can be:

- *observations*: the direct result of the perception processes, representing the environment situation observed by the agent;
- *motivators*: cognitive elements that represent the situations that the agent is trying to achieve, acting as driving forces of the agent behavior, like the motivations;
- *mediators*: cognitive elements that are the resources that support the action, forming an interface between the internal cognitive processing and the concrete action, as illustrated in figure 3.3 taken from [6].

While the observations are the result of the perception activity, motivators and mediators are produced internally, as a result of cognitive activity, or explicitly embedded in the cognitive structure of the agent due to design options or structural restrictions.

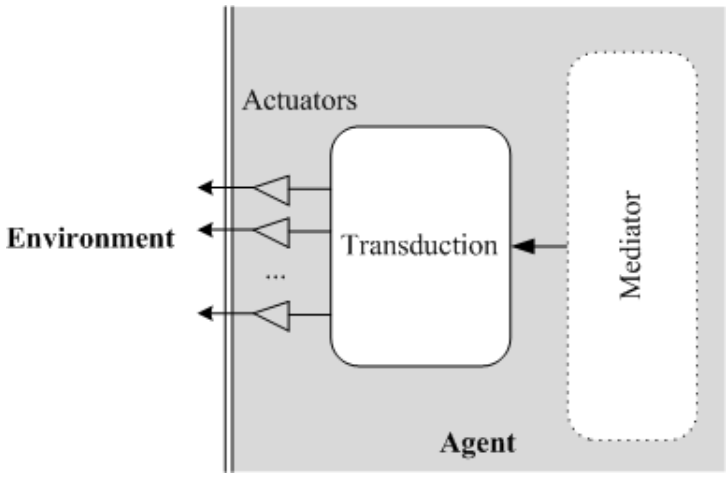

Figure 3.3: Mediators as an interface for concrete action.

Cognitive Activity Periods

In the proposed model, the activity of the cognitive processes occurs during periods of cognitive activity. It is during these periods that the cognitive potentials are generated and interact, producing new cognitive elements, which are considered stable after an initial phase of transition between periods of cognitive activity, as illustrated in figure 3.4.

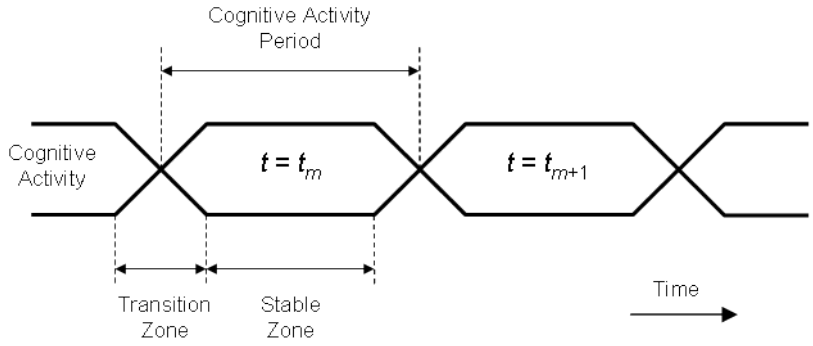

Figure 3.4: Cognitive activity periods evolution along time.

These periods determine the minimum time unit discriminated by the agent. The duration of those periods is inherently not null and result from the agents subjective time characterization into discrete moments t_n with $n \in \mathbb{N}$.

During the stable zone of a cognitive activity period, the characteristics of the existing cognitive elements remain unchanged, although new cognitive elements may be generated. Thus the cognitive elements are seen as localized in time, with an inherently transient existence, unless they are made persistent by assimilation or learning processes, for example, in the form of memories.

Cognitive Space

The cognitive structure allows the agent to keep an internal state that reflects the evolution of the interaction, between agent and environment, over time, named cognitive space.

A cognitive space CS_K is characterized by a set of K base orthonormal vectors, with $K \in \mathbb{N}$, corresponding to quality dimensions, here named cognitive dimensions.

In the cognitive space, cognitive elements can be represented as points. Since the cognitive elements can be localized in time, $t = \tau$, a cognitive element $\sigma(t)$ is represented in the cognitive space CS_K as a vector σ, defined as:

$$\sigma = (\rho_1, \rho_2, ..., \rho_K) \tag{3.3}$$

The topology of a cognitive space is characterized by a metric d that defines the distance between two cognitive elements, σ_1 and σ_2:

$$d(\sigma_1, \sigma_2) \equiv \|\sigma_1 - \sigma_2\| \quad with \quad \|\sigma\| = \sqrt{\langle \sigma, \sigma \rangle} \tag{3.4}$$

where $\|x\|$ represents the norm of vector x and $\langle x, y \rangle$ represents the scalar product between vectors x and y.

To allow the differentiation between different cognitive elements, we will assign unique identifiers to each cognitive element. For example, an agent capable of producing observations from two sensors, a right sensor (RI) and a left sensor (LE), is characterized by two cognitive elements: σ_{RI} and σ_{LE}. Since the cognitive elements represent different locations in the cognitive structure, σ_{RI} and σ_{LE} can also be recognized by their positions.

We should note that the cognitive elements are transient. What this means is that a cognitive element is formed, plays its role in the cognitive activity for a certain period of time, and disappears. If, later on, another cognitive element takes shape in the same location earlier, it is considered a distinct cognitive element. In this sense, the cognitive elements are also located in time. For clarity of notation, this temporal location is implied throughout this report, unless explicitly indicated otherwise.

The concepts presented here are essential because they allow an easy formal way to calculate distances between cognitive elements, corresponding to the level of similarity between those elements. From that distance, we can then calculate, over time, the speed and acceleration between cognitive elements, allowing, for example, to know whether the agent is approaching or departing from its motivations. It is from these dynamics that the emotional phenomena will emerge, following the emotion model presented.

3.1.1 From movement in the cognitive space to emotional dispositions

As the agent interacts with the environment, its cognitive elements will change accordingly, and those changes can be seen as trajectories in the cognitive space. As we have seen, the behavior of an agent is driven by the relationship between the agents motivations and the perception of its current situation, expressed by motivators and observations.

The cognitive activity of the agent is therefore guided by maximizing the flow of achievement that leads to the reduction of the distance between motivators and observations. This process can be described based on the movement that motivators and observations draw in the cognitive space, where the motivators and observations, at a given moment, correspond to specific positions, and the mediators define directions of movement, as illustrated in the figure 3.5 taken from [6].

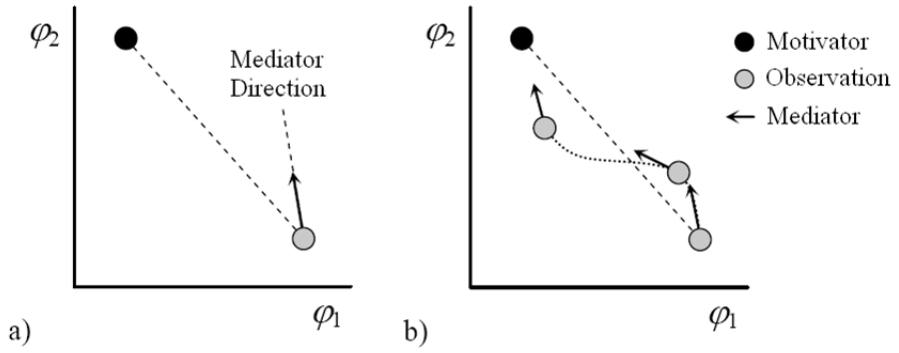

Figure 3.5: The role of a mediator, defining a direction of movement of an observation as the agent acts to try to attain a motivator.

In figure 3.5.b we can see several adjustments to the trajectory of the observation. These could be the result of new behaviors or planning steps, but the forces that led to those changes result from the motivations of the agent and the perception of the evolution of the agent situation in its environment. In the cognitive space these dynamics are expressed by the movement of a observation (obs) in relation to a motivator (mot), depicted in figure 3.6 taken from [6].

The notions of achievement potential and flow are represented, in the cognitive space, by the notions of distance and velocity, because they express the evolution of the motivational achievement of the agent. So, the emotional dispositions (ED) are defined by the evolution of the distance $s = d(\sigma_{obs}, \sigma_{mot})$ and by the velocity $v = ds/dt$ of the movement of σ_{obs} toward σ_{mot}:

$$ED = (\delta_s, \delta_v) \quad where \quad \delta_s = -\frac{ds}{dt} \quad and \quad \delta_v = \frac{dv}{dt} \tag{3.5}$$

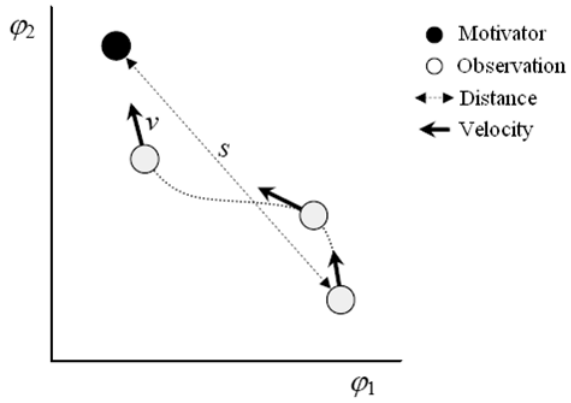

Figure 3.6: Movement of an observation toward a motivator in the cognitive space.

3.2 The Base Mechanisms

After understanding the cognitive structure and how information is represented we will introduce the mechanisms that actually support the creation of emotional dispositions and their use, the base mechanisms.

The base mechanisms, using the cognitive structure, provide basic support to the cognitive activity, regulating and synchronizing the cognitive processes.

3.2.1 Emotional Disposition Mechanism

This mechanism calculates the evolution of the situation between two cognitive elements, a motivator and an observation, producing two types of signals:

- the emotional disposition cognitive potentials, p_s and p_v, that form the emotional disposition vector $ED = (p_s, p_v)$;
- and the affective signals, λ^+ and λ^-, that correspond to the affective property and positive and negative value of an emotional disposition.

The cognitive potentials p_s and p_v belong to two specific cognitive dimensions that, together, represent the emotional disposition space, illustrated in figure 3.7 taken from [6]. Since the emotional disposition is an essential part of the architecture, these two cognitive dimensions are considered implicit to all AutoFocus agents.

Considering the values of the emotional cognitive potentials, they express the base emotional dynamics presented by δ_s and δ_v. We can see in figure 3.7 the emotional disposition vector and how it fits in the emotional patterns previously characterized. For example, when both emotional potentials are positive that represents a positive evolution and concretization perspectives for the agent that can be

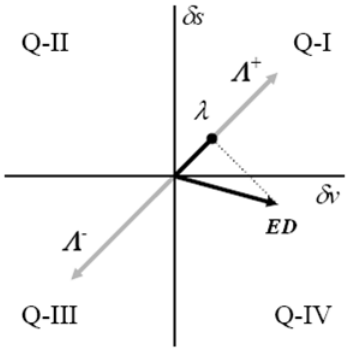

Figure 3.7: The emotional disposition space and an emotional disposition vector.

matched in the emotional pattern of *Joy*, first quadrant of the emotional disposition space.

In this space, the most extreme possible emotional dispositions are represented by the reference vectors $\Lambda^+ = (1, 1)$ and $\Lambda^- = (-1, -1)$, also visible in figure 3.7. It's the projection of the emotional disposition vector on one of those vectors that provides the associated affective values $\lambda \in \mathbb{R}$:

$$\lambda^+ = \begin{cases} proj(ED, \Lambda^+) & , if \quad proj(ED, \Lambda^+) > 0 \\ 0 & , otherwise \end{cases} \tag{3.6}$$

$$\lambda^- = \begin{cases} proj(ED, \Lambda^-) & , if \quad proj(ED, \Lambda^-) > 0 \\ 0 & , otherwise \end{cases} \tag{3.7}$$

where $proj(x, y)$ represents the orthogonal projection of the vector x on the vector y.

The emotional disposition mechanism is the first of the base mechanisms, since the emotional potentials and affective values will serve as input for the other mechanisms.

3.2.2 Regulation Mechanisms

Since the agent time and resources are finite and the need to take action more or less urgent, it will have to confine its cognitive activity. In the proposed model, these two focusing perspectives, of time and resources, are addressed by two base mechanisms, the attention focus and the temporal focus mechanisms. Both of them dependent of the notion of emotional disposition previously described.

The attention focus restricts the accessibility of cognitive processes to the cognitive structure in order to limit the number of cognitive elements available for processing. This way, without altering the cognitive processes it is possible to press the agent for a quicker response by limiting its input.

The temporal focus works through the generation of an indication of the urgency of the response, restricting the time available for the generation of that response.

An observation and a motivator along with cognitive potentials, p_s and p_v, that constitute the associated emotional disposition are integrated to form a more complex cognitive element, σ_{DE}. These integrated elements are then presented to the attention focus mechanism that will decide which ones will be ultimately presented to the cognitive processes.

Attention Focus Mechanism

This mechanism acts like a depletion barrier producing an attention field formed by the integrated elements that are able to cross that barrier. The cognitive processes only have access to the elements in the attention field. Figure 3.8, taken from [6], illustrates this mechanism in action.

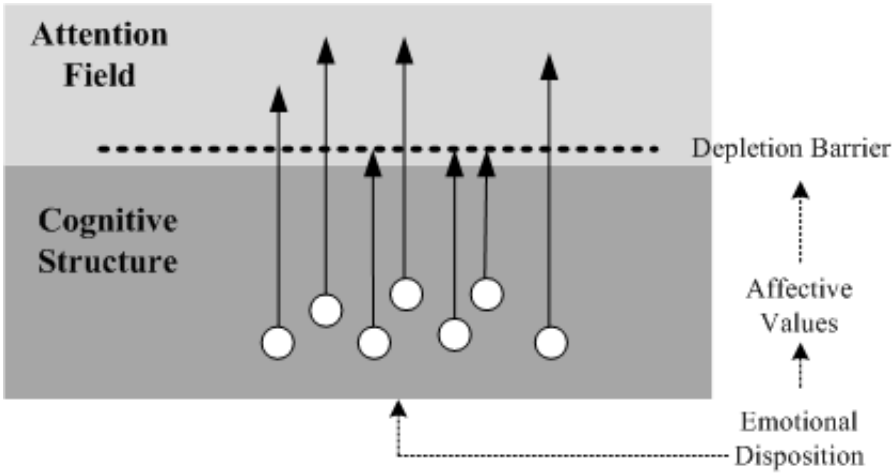

Figure 3.8: Attention focus mechanism.

The depletion barrier is characterized by its intensity and permeability. The depletion intensity ϵ, is regulated by the affective values λ^+ and λ^-, in a way that it can express the cumulative effect of those values:

$$\frac{d\epsilon}{dt} = \alpha^+\lambda^+ + \alpha^-\lambda^- \tag{3.8}$$

where α^+ and α^+ are sensibility coefficients that determine the influence of the affective values, λ^+ and λ^- respectively.

The permeability μ, determines the intensity ϵ^σ of the interaction of the integrated cognitive element σ with the depletion barrier;

$$\epsilon^\sigma = \mu_s p_s^\sigma + \mu_v p_v^\sigma \tag{3.9}$$

where μ_s and μ_v are permeability coefficients that determine the influence of the emotional potentials p_s^σ and p_v^σ of the element σ. If the interaction intensity ϵ^σ is greater than the depletion barrier intensity ϵ ($\epsilon^\sigma > \epsilon$), then the integrated cognitive element σ is included in the attention field.

Temporal Focus Mechanism

The temporal focus mechanism regulates the rate of the cognitive activity. The temporal base corresponds to a signal p_ϕ with a frequency ω_ϕ which can be used to determine the cognitive activity period.

The regulation of the frequency ω_ϕ is determined by the affective values λ^+ and λ^- using the following equation:

$$\frac{d\omega_\phi}{dt} = \beta^+\lambda^+ + \beta^-\lambda^- \tag{3.10}$$

where β^+ and β^- are sensibility coefficients that determine the influence of the affective values, λ^+ and λ^- respectively.

The variable length of the cognitive activity periods, according to the reference time signal p_ϕ, allows an indirect regulation of the type and scope of the processing performed. For example, the perception process can be more detailed or comprehensive depending on the time available for it.

The division of the time available for each cognitive process is relegated to agent designer. It is not a responsibility of this mechanism.

3.2.3 The Base Mechanisms operational view

An operational view of the presented base mechanisms is illustrated in figure 3.9, adapted from [7].

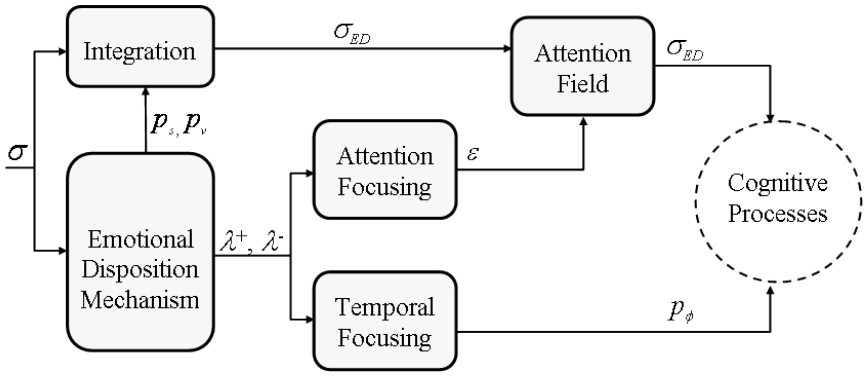

Figure 3.9: Operational view of the presented base mechanisms.

The cognitive elements, an observation and an motivator, are submitted by the emotional disposition mechanism which calculates the corresponding emotional disposition and affective values. These results are integrated along with the initial

cognitive elements to form an integrated cognitive element. After the integration, this element is submitted to the attention focusing mechanism where it's interaction with the attention barrier will be calculated. If that interaction is greater than the attention barrier value, the element is introduced into the attention field.

Meanwhile the affective values are used by the attention focusing mechanism to update the attention field barrier and by the temporal focusing mechanism to update the activity period length.

In the end, only the significant cognitive elements will be presented to the cognitive processes which will have a limited time to compute them in order to calculate the agent's next action.

3.3 The Cognitive Processes

The cognitive processes, as shown in figure 3.1, Perception, Assimilation, Reasoning, Decision and Action, represent generic processes, which may involve several specific processes organized into different levels of detail. There are no restrictions on the form of their implementation inherent to this agent model. What they have in common is the access to the same information, through the attention field, and the possibility to interact with the base mechanisms.

Chapter 4

Design and Implementation of the AutoFocus Platform

This chapter describes the objectives, the decisions and the implementation of the AutoFocus platform as a *Java* library.

4.1 Objectives, Planning and Methodology

This project had two main objectives: to clarify some aspects of the Agent Flow Model and to provide the necessary tools to facilitate the development of agents with this architecture.

Although the theory of the Agent Flow Model had been developed, as it was presented in the previous chapter, it had only been implemented and tested for specific cases. Until the beginning of this project several prototypes existed, but both the cognitive structure and the base mechanisms had been specifically implemented for each one according to the problem addressed. Those prototypes were implemented in the *C* language, with more emphasis in efficiency than in generality.

So, this project was idealized to integrate the knowledge gathered from each prototype and to provide a general reusable implementation of both the cognitive structure and the base mechanisms, the key features of the Agent Flow Model.

This implementation, as determined in the project specification, would take the form of a *Java* library.

Planning

Given the complexity of the agent model and the uncertainty that still existed about certain aspects, it was decided to adopt a iterative approach to the development effort, dividing it into two main iterations.

In the first iteration the base mechanisms including the regulation mechanisms

were to be implemented as well as prototypes to test them. The second iteration would then be devoted to develop extensions to those base mechanisms to allow the implementation of the memory mechanisms and prototypes with memory formation.

It was decided that the first iteration should be concluded in mid-March and the second iteration in early July, as depicted in figure 4.1.

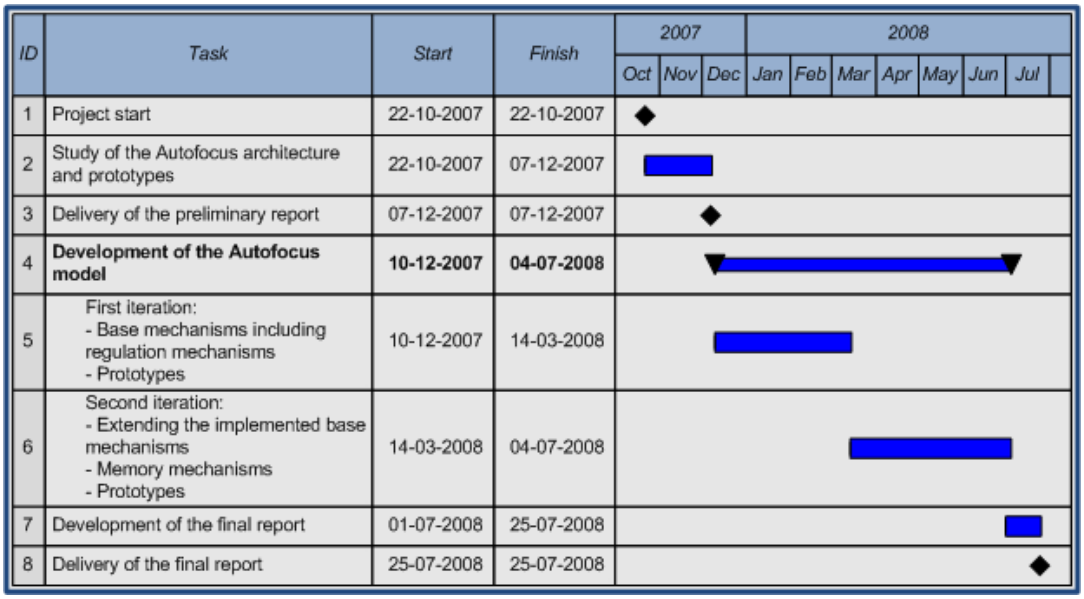

Figure 4.1: The planning presented in the preliminary report.

Actually the planning has not been completely fulfilled and the second iteration was relegated to future work. This happened for two reasons. First, the complete design and implementation of the base mechanisms took longer than expected. Second, a lot of effort was put in the development of a stable platform to test the prototypes. This last task was not planed but it was considered to be an important part of the AutoFocus platform.

Some initial work was done for the second iteration, for example on the domain model, which will be presented in this report, although it was not implemented.

Methodology

The methodology adopted was to have weekly meetings, with the supervising professors, Prof. Graça Gaspar and Prof. Luís Morgado, to discuss the concepts of the model and define the domain model and the general conception of the platform. In each meeting the work done so far was presented and evaluated. When the design and implementation phases started, the meetings frequency was reduced to fortnightly and focused more on configuration options and the experimentation aspects of the AutoFocus platform.

There was an effort to follow the unified modeling process during the design and implementation of this platform.

4.2 General Platform Conception

Using the Rl-Glue[9] as source of inspiration, it was decided to divide the platform into three subsystems: the Agent, the Environment and the AutoFocusGlue.

The Rl-Glue is a standard for connecting reinforcement learning agents and environments.

In theory, the RL-Glue is a protocol consisting of standard functions to facilitate the exchange and comparison of agents and environments without limiting their abilities. As software, RL-Glue is functionally a test harness to "plug in" agents, environments and experiment programs without having to continually rewrite the connecting code for these pieces.

Using this approach in the Agent Flow Platform architecture lead to the creation of the AutoFocusGlue.

The AutoFocusGlue

The AutoFocusGlue was introduced to control the communication between the agent and the environment. The objective was to facilitate the decoupling between agents and environments, so that different agents could be easily tested with different environments and vice-versa.

It also allows the execution of the agent and environment systems at different rates of activation.

The Agent System

The Agent system is composed by the cognitive structure, the base mechanisms and the cognitive processes, following the architecture of the Agent Flow Model (figure 3.1).

However there are some important differences. The perception and action processes were relegated to the environment system implying that the agent system is liberated from the responsibility of transforming sensory input into cognitive elements or transforming action mediators into the actual physical execution of that action on the environment.

What this means is that the perception, along with all it's steps, is done by the environment which transmits the appropriate observation cognitive elements to the agent.

Using the same perspective, after the agent computes what will be it's next action, it sends the corresponding cognitive elements to the environment which then makes the necessary changes to the environment and agent states.

So the agent system can best be described as the "mental" side of the agent, while it's "physical" characteristics exist and are manipulated by the environment system.

The Environment System

The environment system is responsible for maintaining the environment and agent physical states. The environment is also responsible to provide the agent with a cognitive space specification powerful enough for the agent to interact with the environment.

When queried about it's current state the environment must perform the transduction and manipulation, inherent to the perception, necessary to provide the agent with an observation that corresponds to the agent's view of the environment. That observation must be in accordance to the defined cognitive space, the reality perceptible by the agent. When it is requested for the environment to execute an agent's action, it must transform the action into the proper agent and environment modifications.

4.3 Domain Model

The agent domain model is presented in figure 4.2, followed by a description of every entity.

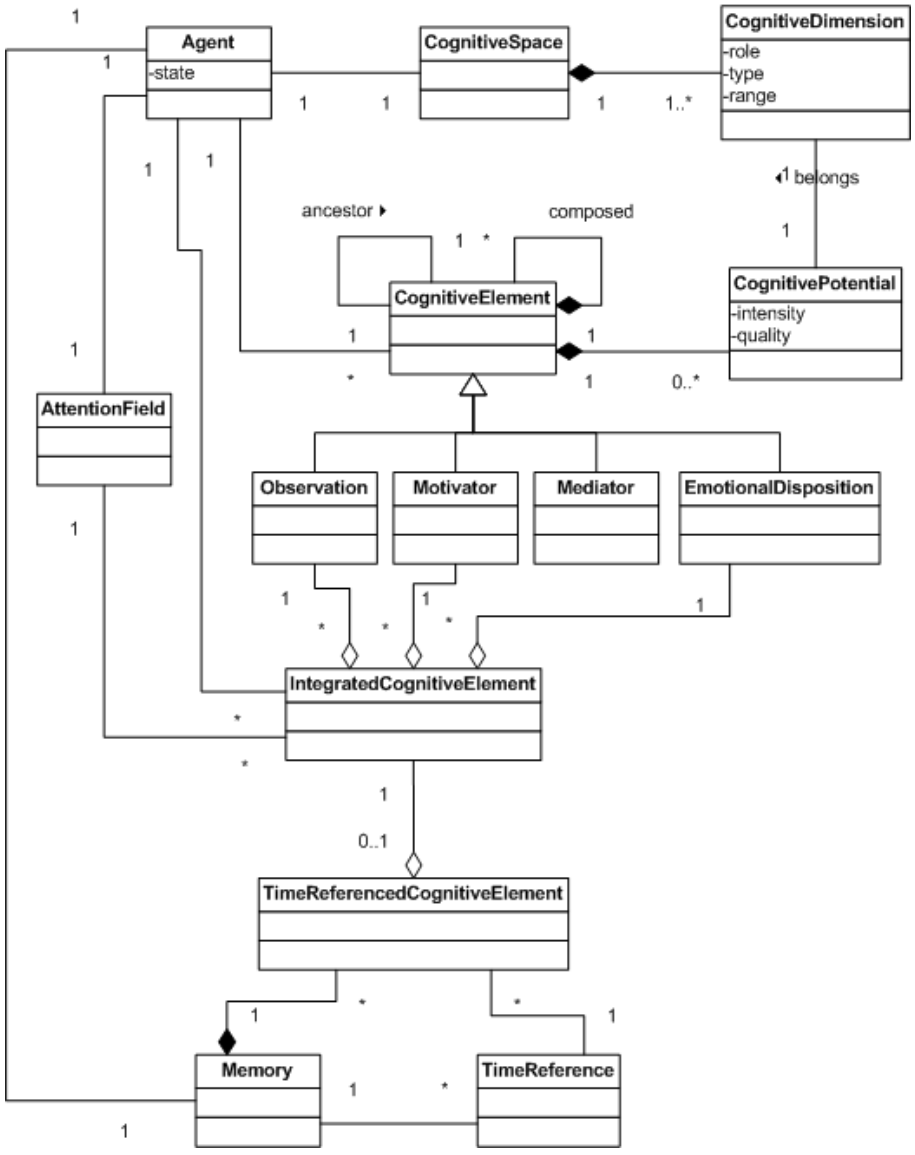

Figure 4.2: The AutoFocus agent domain model.

Agent

This is the main entity of the Agent system, representing the Agent. It has a state that will be used to ensure a proper functioning of the system. The agent mainly associated to a cognitive space since it determines the agent's view of the reality.

CognitiveSpace

This entity represents the cognitive space, which is composed by one or more cognitive dimensions.

CognitiveDimension

A cognitive dimension is characterized by it's type and range of values and it's role in the cognitive space.

The dimension's type and range restrict the acceptable values. For example, a cognitive dimension could accept only integers from 0 to 10. It is also possible to have user defined value types as long as those values are ordered and a function is defined that can calculate the distance between any two of those values.

The role of the cognitive dimension was introduced to facilitate the discrimination between the cognitive dimensions that compose the observations from those that compose the mediators.

CognitivePotential

The cognitive potentials are defined by a quality and an intensity. Each cognitive potential belongs to a specific cognitive dimension. That relation restricts the potential quality value to the type and range of the cognitive dimension. The intensity, on the other hand, is always a real number in the interval $[0, 1]$.

CognitiveElement

The cognitive element is composed either by a set of one or more cognitive potentials or by a set of one or more cognitive elements, in which case it is called a composed cognitive element.

Since cognitive elements evolve with time, each cognitive element has a link, *ancestor*, which indicates its predecessor, i.e. the cognitive element corresponding to itself in the immediately preceding moment of time. This connection is necessary to establish the trajectory of a cognitive element in the cognitive space.

The cognitive elements can be classified by their role in the cognitive structure: observation, motivator, mediator and emotional disposition. The emotional disposition can be considered a cognitive element since it is composed by two cognitive potentials, according to the emotional model used in this project.

IntegratedCognitiveElement

The instances of this entity will be created by the emotional disposition mechanism and are composed by an observation, a motivator and the emotional disposition that results from their interaction.

AttentionField

It represents the set of integrated cognitive elements that are available to the cognitive processes.

TimeReferencedCognitiveElement

It is an integrated cognitive element that can be referenced in time.

TimeReference

Represents a time reference used by the agent.

Memory

It consists of the integrated cognitive elements, referenced in time, that were memorized by the agent.

4.4 The Agent, Environment and AutoFocusGlue Systems

As it was previously presented, the AutoFocus platform is composed by three systems: the Agent, the Environment and the AutoFocusGlue, illustrated in figure 4.3. The main objective for this architectural decision was to provide a flexible platform, facilitating the running of sets of experiments.

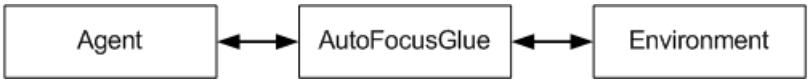

Figure 4.3: The three systems that compose the AutoFocus platform.

With a general communication interface between the three systems it is possible to test the same agent with different parameters, to study it's performance, or to test different agents with the same environment and vice-versa. To achieve this, both the Agent and the Environment have several specific functions which will provide a well established interface. These will be presented in the rest of this section.

Note that the Agent and Environment functions should, ideally, only be called by the AutoFocusGlue, and not directly, if one wants to use AutoFocusGlue to control experiments and obtain and treat experimentation results.

I will first present the available functions of the Agent and Environment systems and after that the AutoFocusGlue functions and how it communicates with the Agent and Environment to ensure a proper execution of the platform.

4.4.1 The Agent System

The agent has four possible states: *created*, *initiated*, *active* and *suspended*. These states and their sequence are depicted in figure 4.4.

The agent interface is composed by three main functions: *init* (initialization), *start* and *step*.

init(agentInit, css, distFunction)

- *agentInit* - The agent's initialization parameters.
- *css* - The cognitive space specification.
- *distFunction* - The distance algorithm associated to the cognitive space.

This function is used to initiate or re-initiate the agent. The agent's parameters are reset to their initial values and the cognitive space is built, or rebuilt, according to the cognitive space specification. This implies that all existing cognitive elements

are erased. The agents distance algorithm, which calculates the distance between cognitive elements, is set according to the parameter *distFunction*.

The base mechanisms are also initiated, and their parameters reset to their initial values.

The agent system state is set to *initiated*.

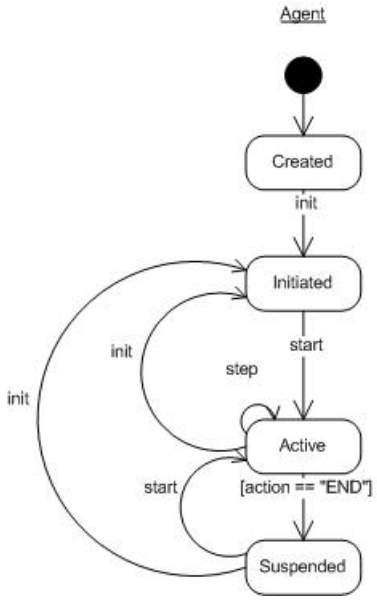

Figure 4.4: The Agent system states.

start(agentConfig, agentExec, goal, goalFunction)

- *agentConfig* - Agent configuration parameters.
- *agentExec* - Agent execution parameters.
- *goal* - The initial agent goal (motivator).
- *goalFunction* - The goal function.

This function can only be called when the agent system is in the *initiated* state.

The agent's configuration and execution parameters are set, as well as the agent's initial motivator and goal function. The configuration parameters are related to the base mechanisms variables, and will be presented in the next section. The execution parameters where introduced to allow the agent's designer to set context specific values. The goal function is only necessary when the agent's built-in motivation cannot be expressed by a static motivator. This is will be explained ahead, in the messages definition section.

The agent system turns to the *active* state.

step(envState) - action, timeTaken

- *envState* - The current environment state.

This function can only be called when the agent system is in the *active* state.

This is the main function of the agent computation. It takes the current environment state, in the form of an observation, and computes the next action. What exactly are the individual steps and mechanisms used, will be described in detail in the next section.

It returns the action and the time taken to compute it. The time taken is the subjective time that the agent took to compute the action. This time is constrained by the temporal focus configuration parameters, that set its maximum value. But the time taken can also be less than that specified limit and that is why it is returned by the *step* function.

If the action returned is "END", signaling that all motivations have been achieved, the agent state changes to *suspended*, if not, it continues *active*.

4.4.2 The Environment System

The environment subsystem also has four states: *created*, *initiated*, *active* and *suspended*. It's interface is composed by four main functions: *init*, *start*, *evolve* and *executeAction*. Figure 4.5 illustrates the environment states and their sequence.

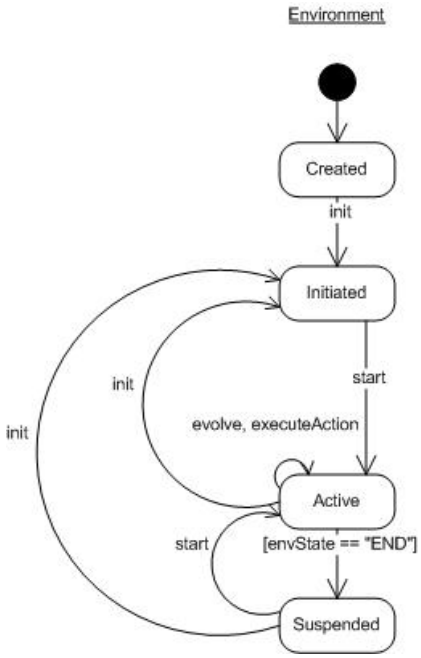

Figure 4.5: The Environment system states.

init(envInit) - css

- *envInit* - The environment's initialization parameters.

When this function is called, all the environment parameters are set or reset to their initial values and the environment is initiated or re-initiated. This function returns the cognitive space specification, representing the cognitive dimensions that are needed for an agent to interact with the environment.

The environment system changes to the *initiated* state.

start(envExec) - envState

- *envExec* - The environment's execution parameters.

This function can only be called when the environment system is in the *initiated* state.

This function sets the execution parameters and starts the environment execution. It returns the current environment state, which in this case is the initial observation.

The environment system evolves to the *active* state.

evolve() - envState

This function can only be called when the environment system is in the *active* state.

The environment should evolve according to its internal dynamics and return the resulting environment state.

If the resulting environment state is different from "END", which signals the end of the environment evolution, the environment system remains in the *active* state, otherwise it turns to the *suspended* state.

executeAction(action) - envState

- *action* - The agent's action.

This function can only be called when the environment system is in the *active* state.

The environment executes the agent's action by modifying accordingly the environment and agent representations. It returns the resulting environment state.

If the resulting environment state is different than "END", which signals the end of the environment evolution, the environment system remains in the *active* state, otherwise it turns to the *suspended* state.

4.4.3 The AutoFocusGlue System

The AutoFocusGlue system has three states (*suspended*, *initiated* and *active*) and three main functions (*init*, *start* and *step*), illustrated in figure 4.6.

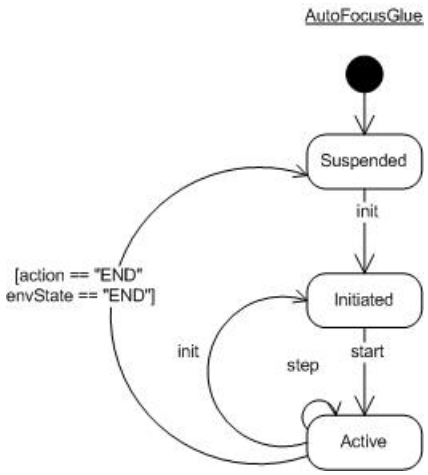

Figure 4.6: The AutoFocusGlue system states.

init(envPeriod, envInit, agentPeriod, agentInit)

- *envPeriod* - Environment activation period.
- *envInit* - Environment initialization parameters.
- *agentPeriod* - Agent activation period.
- *agentInit* - Agent initialization parameters.

This function stores the activation periods and initializes, first the environment and then the agent, as depicted in figure 4.7. It also initializes the time ticker. The agent's and environment's activation periods are integers whose value determine the relative rate of activation of these two subsystems. An environment period of 100, for instance, means the environment should evolve once for each 100 ticks of the AutoFocusGlue counter.

The AutoFocusGlue system changes to the *initiated* state.

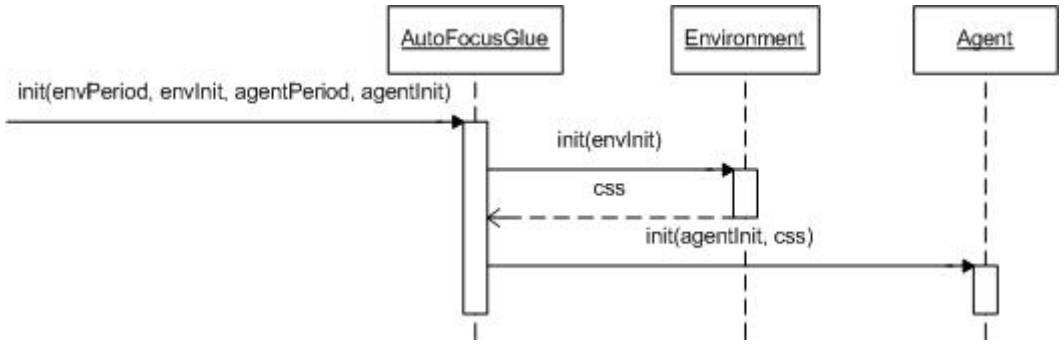

Figure 4.7: The AutoFocusGlue initialization sequence.

start(envExec, agentConfig, agentExec, goal, goalFunction)

- *envExec* - Environment execution parameters.
- *agentConfig* - Agent configuration parameters.
- *agentExec* - Agent execution parameters.
- *goal* - Agent initial goal or motivation.
- *goalFunction* - Agent goal function.

This function can only be called when the AutoFocusGlue system is in the *initiated* state.

This function starts the execution of the agent and then the environment, storing the returned environment state, as depicted in figure 4.8.

The AutoFocusGlue system evolves to the *active* state.

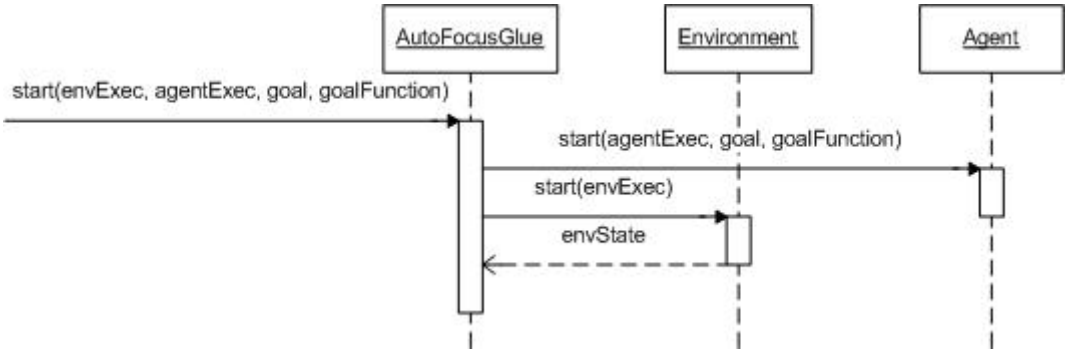

Figure 4.8: The AutoFocusGlue start sequence.

step()

This function can only be called when the AutoFocusGlue system is in the *active* state.

This function makes the computation advance "one step", as depicted in figure 4.9. Here, the expression "one step" is used to refer to the computation of the agent's next step and the corresponding evolution of the environment.

It starts by calling the step function of the Agent, with the previously stored environment state as parameter. The Agent responds with the next action and the time taken for its computation.

To represent the different activation rates, and the time taken by the agent to determine an action due to the temporal focusing, the AutoFocusGlue will determine how many times the Environment must evolve before the agent's action take place.

The algorithm that defines how many times the environment evolves in relation to the agent is also illustrated in figure 4.9.

The AutoFocusGlue maintains two internal variables, *agentLastEvolved* and *envLastEvolved*, that record when, in terms of tick counts, the agent and the environment were last invoked. The next time the agent should act upon the environment depends on the *agentPeriod* and on the *timeTaken*, i.e. the subjective time the agent took to compute that action. The variable *timeTaken* is seen as a number of units of *agentPeriod*s.

If either the resulting action or environment state are "END" the AutoFocusGlue changes to the *suspended* state otherwise it stays in the *active* state.

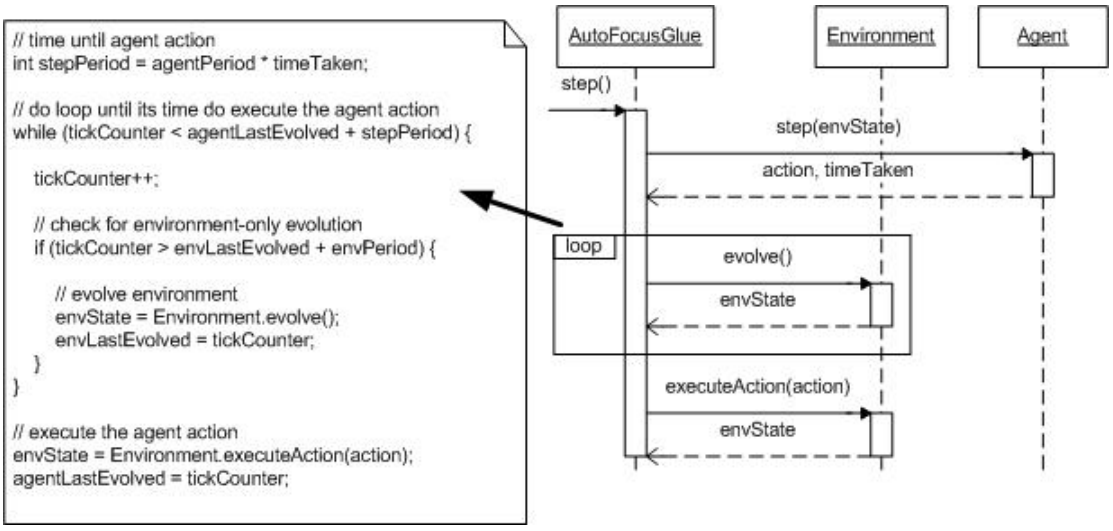

Figure 4.9: The AutoFocusGlue step sequence and algorithm.

4.4.4 Messages Definitions

The messages presented previously on the AutoFocus interfaces show a great deal of information being passed between the agent and the environment. This section will introduce the syntax and semantic of those messages.

Upon initialization the environment sends a CSS (cognitive space specification) message to the agent.

Cognitive Space Specification (CSS)

The CSS indicates the cognitive dimensions that compose the cognitive space. It is composed by: the total number of dimensions, the specification of the observation dimensions and the specification of the action dimensions.

```
totalNumber|obsDimensions|actDimensions
```

Both the observation and action dimensions are described by their type and an optional range of allowed values:

```
type1range1:type2range2:...:typeNrangeN
```

The type indicates the type of value of that particular dimension. For the moment, only two types are implemented, integers and doubles, indicated in a CSS description by `i` and `d` respectively. Expansion to other types, including user designed types has also been considered and the implementation was chosen to facilitate that, as will be mentioned in section 4.5.1.

The range specifies the interval of values allowed in that dimension, $[min, max]$. If min or max are not present, the type limits are assumed, for example, in Java an integer has a minimum value of -2,147,483,648 and a maximum value of 2,147,483,647.

Here is an CSS example:

```
4|i[0,1]:d[,]:d[1.5,3]|i[0,3]
```

This example has four dimensions, three observation dimensions and one action dimension.

Cognitive Potential and Cognitive Element Representation

The cognitive potential, as presented before, is composed by two values: intensity and quality. In the messages exchanged we will refer to it's quality value simply as value, while the intensity value will be referred as it's intensity.

The cognitive potential is always bound to a cognitive dimension that indicates the type and range of the cognitive potential value, while the intensity is always a real number, between 0 and 1.

The cognitive potentials have the following syntax:

```
value,intensity
```

For example, a cognitive potential with value 3.14 and intensity 0.5 is represented by:

```
3.14,0.5
```

By aggregating several cognitive potentials we get a cognitive element.

```
v1,i1:v2,i2:...:vn,in
```

Here is an example of a cognitive element with a value 1 and intensity 1 in the first dimension, value 3.14 and intensity 0.8 in the second dimension and value 11 and intensity 1 in the third dimension.

```
1,1:3.14,0.8:11,1
```

For simplicity sake, when the intensity is 1 it can be omitted. So the above cognitive element could be rewritten has:

```
1:3.14,0.8:11
```

If the potential value corresponding to a certain dimension is not present, the intensity value is assumed 0 and can also be omitted from the description of the cognitive element. For example, if the second dimension of the previous example had no value, then its intensity would be 0 and the description of the whole cognitive element would be as follow:

```
1::11
```

Note that the cognitive elements must respect the format given by the CSS. For example to represent an observation as a cognitive element, its cognitive potentials must match the number, type and range of the specified observation dimensions.

Environment State Representation

Note that the term environment state is used here to refer what the agent observes, as it was explained previously. As such, this state representation can depict, and in most cases it does, only a part of the entire environment state.

In simple cases, an environment state representation can consist of only one cognitive element, i.e. a simple observation, but for more complex situations it is conceived that an environment state can be composed by more than one cognitive element. So there is a need for those elements to be tagged with an object class and unique identification, using the following syntax:

```
objectClass|objectID|observation
```

for a single cognitive element. If there are various cognitive elements they must be separated by the symbol "-":

```
class|ID1|obs1-class|ID2|obs2-...
```

Action Representation

The representation of actions is composed by cognitive elements just like the observations. Note that the actions, in this framework, are atomic. This means that the environment can always execute any of the agents actions in one of its steps.

Goal Representation

A goal or motivator follows the same syntax. If a dynamic motivator is needed, when the motivation cannot be represented by a static motivator throughout the agent life but it rather depends on the objects existing in the current environment state, then each time the agent receives an environment state, there is a goal function that determines the motivators. This function should be defined by the agent developer.

A simple example: The Cat and Mouse World

In the Cat and Mouse World the environment consists of a 10x10 grid world. In this world the agent, which is a cat, must catch one of three mice. This world is depicted in figure 4.10, where C is the cat and M1, M2 and M3 are the mice.

To represent this world in a cognitive space we will use three cognitive dimensions representing the environment cells:

- row number; from 1 to 10
- column number; from 1 to 10
- cell content;
  - 0, empty
  - 1, cat
  - 2, mouse

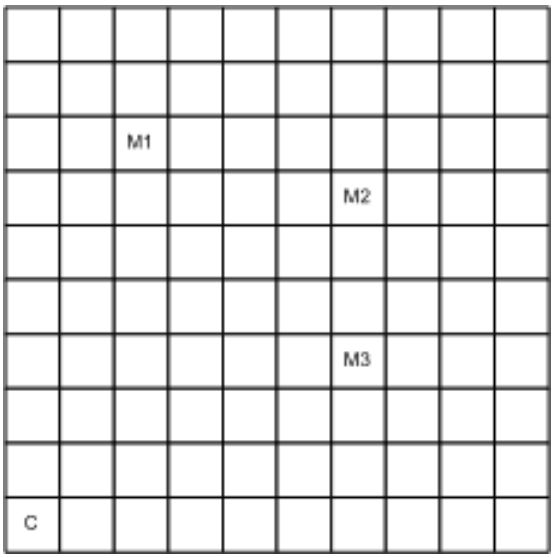

Figure 4.10: The Cat and Mouse World.

– 3, cat and mouse

The cat can perform the following two types of action: move to an adjacent cell; and catch a mouse, if there is one at his current position. To represent these actions we will use two dimensions. One indicates a movement action and the other a catch action.

- movement
  - 1, move to the upper left cell
  - 2, move to the upper cell
  - 3, move to the upper right cell
  - 4, move to the left cell
  - 5, don't move
  - 6, move to the right cell
  - 7, move to the down left cell
  - 8, move to the down cell
  - 9, move to the down right cell
- catch mouse
  - 0, don't catch mouse
  - 1, catch mouse

So this world CSS, using our approach, is:

```
5|i[1,10]:i[1,10]:i[0,3]|i[1,9]:i[0,1]
```

Using this CSS, we can represent the situation presented in figure 4.10 with the following environment state, composed by four observations:

```
cat|C|10:1:1-mouse|M1|3:3:2-mouse|M2|7:4:2-mouse|M3|7:7:2
```

A movement of the cat to the right cell is represented by the following cognitive element corresponding to a mediator:

```
6:0
```

4.5 Design and Implementation of the Agent System

In this section I will present the class models of the most relevant parts of the AutoFocus platform and the algorithm implemented by the base mechanisms.

4.5.1 From the Domain Model to the Class Model

As it was already explained, one of the most important notions of this agent architecture is the cognitive dimension which is characterized by it's type and range of values.

These specific type of values follows two rules: the values are ordered and there is a distance function to calculate the proximity between any two values.

These restrictions were implemented by imposing that any new type of values must implement the *AutoFocusComparator*, illustrated in figure 4.11, along with the two currently implemented types, Integer and Double.

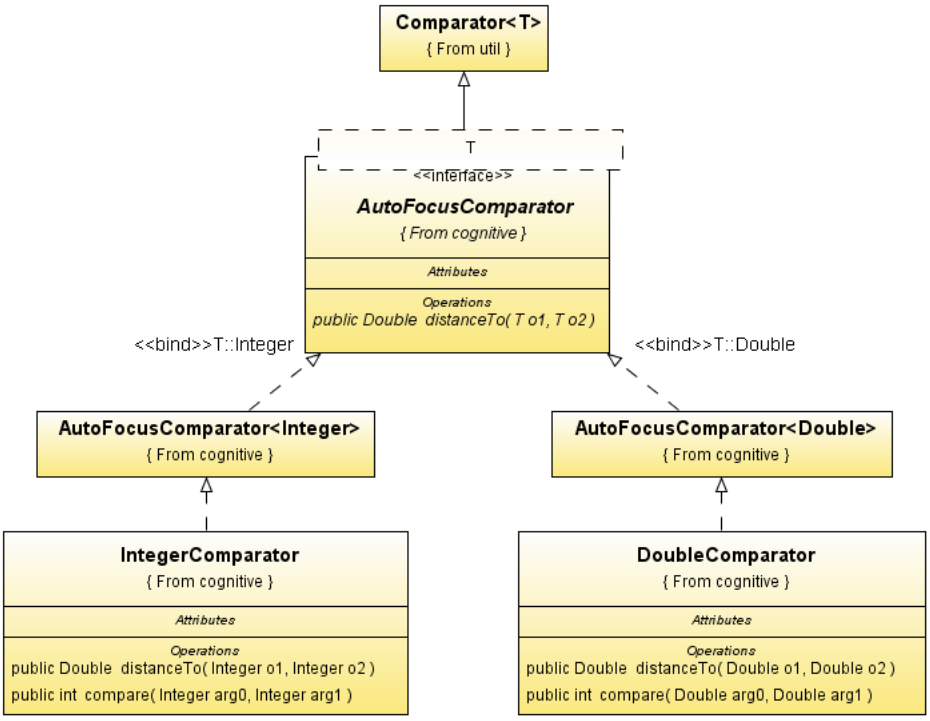

Figure 4.11: The AutoFocusComparator and the two available types.

The cognitive structure class model

By separating the restrictions on the value type of cognitive dimension we have a single generic implementation of cognitive dimension, illustrated in figure 4.12.

There we can see that a generic cognitive dimension is characterized by it's minimum and maximum values, a string describing the value type and a char that identifies it's role in the cognitive space.

In this figure we can also see that a cognitive potential only exists if connected to a cognitive dimension and it's quality is restricted to the type and range of that dimension.

The cognitive space class is composed by two ordered sets of cognitive dimensions. One corresponding to the observation subspace and the other to the action or motivator subspace.

The cognitive elements are composed by an ordered list of cognitive potentials. The cognitive elements also have an association to the cognitive space which imposes that the cognitive potentials follow the description of the corresponding cognitive dimensions.

The integrated cognitive elements class is composed by an observation, an motivator and the emotional dispostion and associated values that result from the emotional and integration base mechanisms.

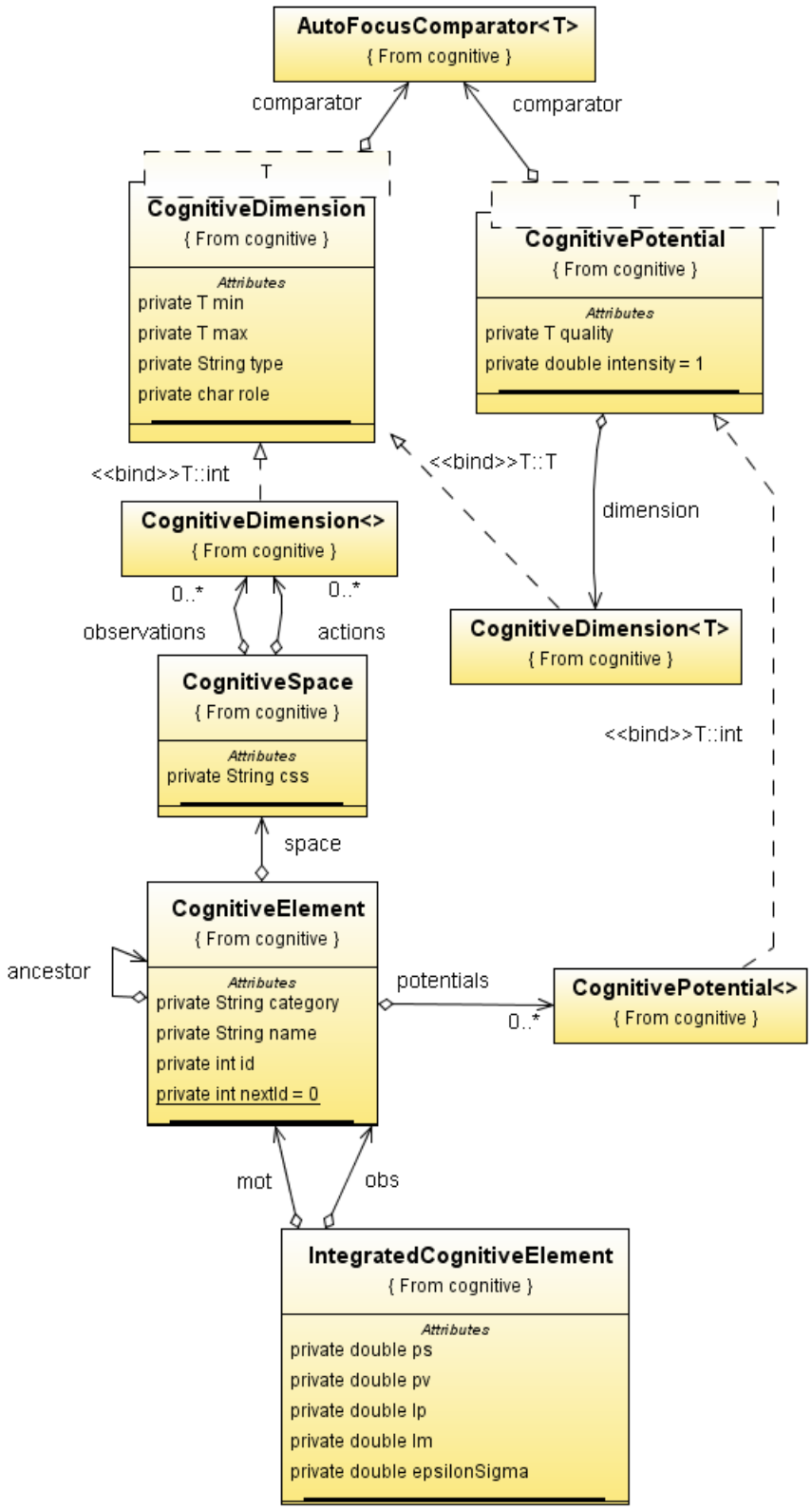

Figure 4.12: A view of the cognitive structure class model.

The Agent class model

The class model view relative to the agent is illustrated in figure 4.13. Here we can see the Agent interface which specifies the agent's communication functions, as presented in section 4.4.

The Agent interface is implemented through the DefaultAgent class, which is characterized by the cognitive space, base mechanisms and the observations, motivators and mediators known to the agent. These cognitive elements are stored through an instant memory class which keeps the current version of any cognitive element and it's ancestor, the necessary information to calculate the emotional dispositions.

The BaseMechanism class stores all the base mechanisms as well as the agent's attention field. It provides the agent with a method that computes a cognitive element pair through all the base mechanisms. This algorithm will be presented in the next section.

Both the default agent and the base mechanisms classes extend the Observable class for monitoring purposes.

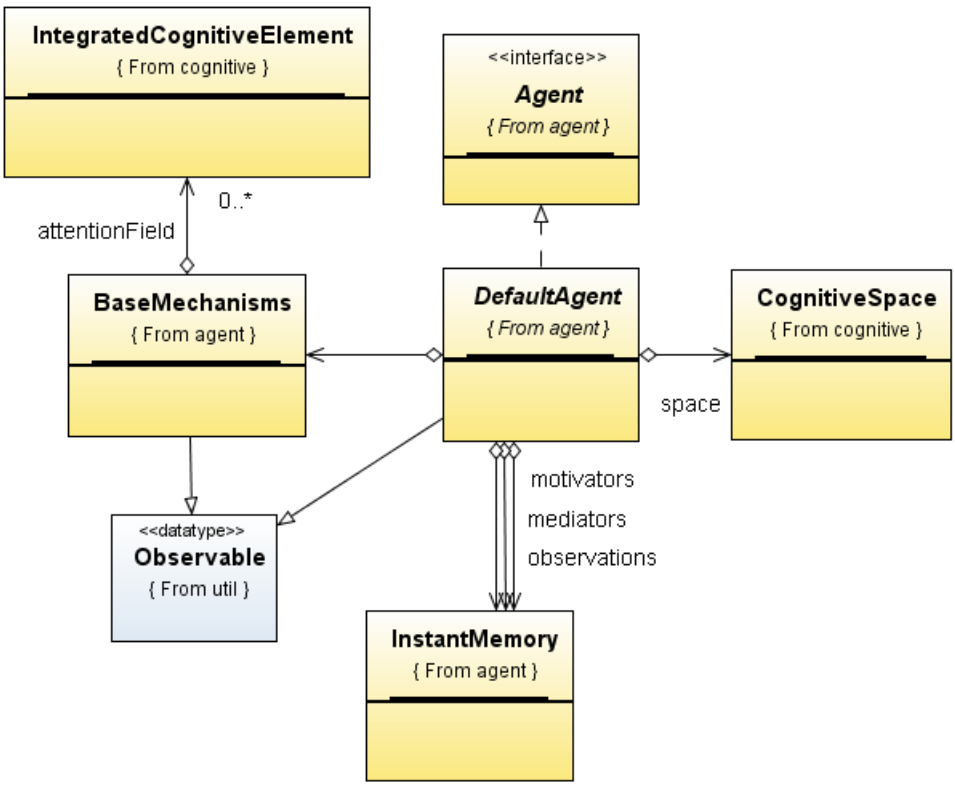

Figure 4.13: The class model of the agent system.

The three systems class model

The three systems that compose the AutoFocus platform (Agent, Environment and AutoFocusGlue) are depicted in figure 4.14. These were first defined as interfaces to implement the communication functions presented in section 4.4. Each interface was then implemented by a class that also extends the *Observable* class for monitoring purposes.

While the AutoFocusGlue default class is completely implemented, the agent's and environment's default classes still have some abstract methods. These two classes provide the basis to any developer that wants to implement an agent or environment in this platform.

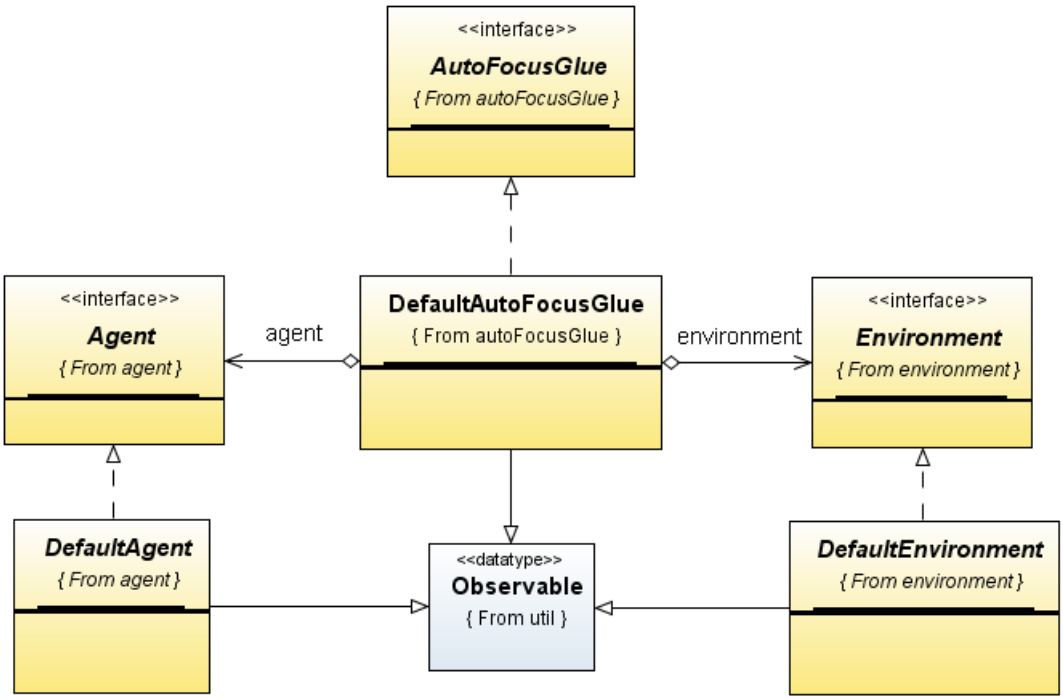

Figure 4.14: The class model of the AutoFocus platform.

4.5.2 The Base Mechanisms algorithm

After a thorough study of the base mechanisms theory and analysis of the existing prototypes, the complete functional computation of a pair (observation, motivator), using all the base mechanisms, was specified and is illustrated in figure 4.15.

In this figure the individual base mechanisms are represented by rounded boxes with their left border crossed by small boxes indicating their input parameters and the resulting outputs represented across the right border. There are also two darker areas indicating the regulation mechanisms and the memory mechanisms.

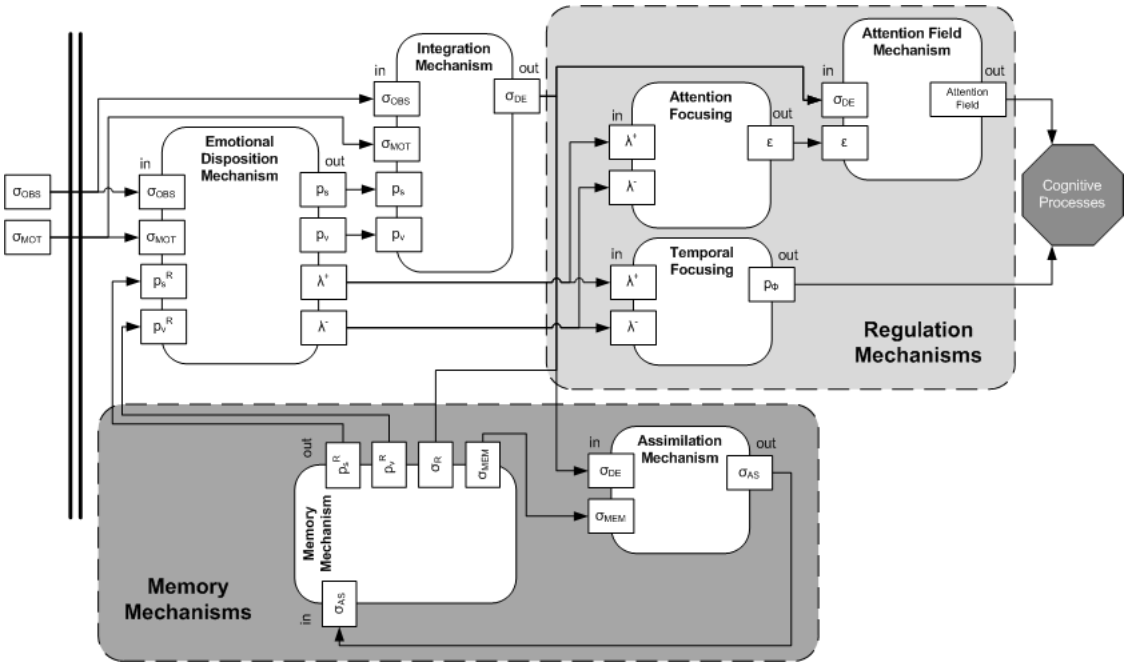

Figure 4.15: Functional view of the base mechanisms.

The memory mechanisms were not introduced in the algorithm that follows since they were not implemented, but it is possible to view in the figure how they would fit in the overall scheme.

Each time the agent receives a new environment state, in the form of one or more cognitive elements, it discriminates those elements into two sets, observations and motivators, which are then processed by the base mechanisms.

Emotional Disposition Mechanism

For each pair of one observation and one motivator (σ_{obs}, σ_{mot}), it is calculated the current and previous distance, s and s^{t-1}, between σ_{obs} and σ_{mot}:

$$s = d(\sigma_{obs}, \sigma_{mot}) \tag{4.1}$$

$$s^{t-1} = d(\sigma_{obs}^{t-1}, \sigma_{mot}^{t-1}) \tag{4.2}$$

If either the observation or motivator does not have an ancestor, for example at the beginning of the agent's execution, then the previous distance s^{t-1} to consider should be defined by default by the distance algorithm. It is typically a maximum value.

From these distances it is calculated the emotional disposition potentials, p_s and p_v:

$$p_s = s^{t-1} - s \tag{4.3}$$

$$p_v = p_s - p_s^{t-1} \tag{4.4}$$

The Integration Mechanism

Combining the observation and motivator, together with the corresponding emotional potentials we can determine their interaction with the attention field barrier ϵ^σ:

$$\epsilon^\sigma = \frac{p_s \times \mu_s + p_v \times \mu_v}{\sqrt{{p_s}^2 + {p_v}^2}\sqrt{{\mu_s}^2 + {\mu_v}^2}} \tag{4.5}$$

The variables μ_s, μ_v represent the permeability of the attention depletion barrier, and have, by default, the values: $\mu_s = 1, \mu_v = 1$. This particular equation is a normalized version of the equation presented previously in section 3.2.2.

After that the lambda values for this integrated element are calculated:

$$\lambda_\sigma^+ = \begin{cases} \frac{p_s + p_v}{\sqrt{{p_s}^2 + {p_v}^2}\sqrt{2}} & , if(p_s + p_v) > 0 \\ 0 & , otherwise \end{cases} \tag{4.6}$$

$$\lambda_\sigma^- = \begin{cases} -\frac{p_s + p_v}{\sqrt{{p_s}^2 + {p_v}^2}\sqrt{2}} & , if(p_s + p_v) < 0 \\ 0 & , otherwise \end{cases} \tag{4.7}$$

The resulting information (the observation, the motivator, the emotional potentials p_s and p_v, the attention barrier interaction ϵ_σ and the two lambda values λ_σ^+ and λ_σ^-) is stored in a new entity called an integrated cognitive element.

After all the integrated cognitive elements have been created, the global affective values, λ^+ and λ^- are calculated:

$$\lambda^+ = \frac{1}{n} \sum_{i=1..n}^{n} \lambda_\sigma^+ \tag{4.8}$$

$$\lambda^- = \frac{1}{n} \sum_{i=1..n}^{n} \lambda_\sigma^- \tag{4.9}$$

where n is the number of current integrated cognitive elements.

The Attention Focus Mechanism

The integrated cognitive elements created are now tested so see if they can reach the attention field or not. Only the integrated cognitive elements whose interaction with the attention barrier is higher than the actual barrier limit are accepted.

$$\epsilon^{\sigma} > \epsilon \tag{4.10}$$

By default, the initial ϵ value is 0.

Note that the attention field is always emptied when the agent receives a new environment state, so there is a possibility that none of the new integrated elements is of significant importance to be in the attention field. To resolve this issue it was introduced a new agent parameter, *emptyAttention*, that signals if the attention field can be left empty or if at least one element, the one with the best interaction, should be included in the agent attention field, if it were to be otherwise empty.

After populating the attention field, the attention barrier value is updated:

$$\epsilon = \epsilon^{t-1} + \alpha^{+}\lambda^{+} - \alpha^{-}\lambda^{-} \tag{4.11}$$

where the variables α^{-} and α^{+} represent the sensibility of the attention barrier to the global affective values. They are constricted to the interval $[-1, 1]$, and have the following values by default: $\alpha^{-} = 0.3, \alpha^{+} = 0.3$.

The attention barrier value is constrained to it's limits, if necessary:

$$\epsilon \rightarrow \begin{cases} \epsilon_{min} & , if \epsilon < \epsilon_{min} \\ \epsilon_{max} & , if \epsilon > \epsilon_{max} \\ \epsilon & , otherwise \end{cases} \tag{4.12}$$

By default the limits of the attention barrier value are: $\epsilon_{min} = -1, \epsilon_{max} = 1$.

The Temporal Focus Mechanism

Now the temporal focus value is also updated:

$$\omega = \omega^{t-1} + \beta^{+}\lambda^{+} - \beta^{-}\lambda^{-} \tag{4.13}$$

where the variables β^{-} and β^{+} represent the sensibility of the temporal focus to the global affective values. They are constricted to the interval $[-1, 1]$, and have the following values by default: $\beta^{-} = 0.5, \beta^{+} = 0.5$.

The temporal focus is constrained to it's limits, if necessary:

$$\omega \rightarrow \begin{cases} \omega_{min} & , if \omega < \omega_{min} \\ \omega_{max} & , if \omega > \omega_{max} \\ \omega & , otherwise \end{cases} \tag{4.14}$$

By default, the initial focus value is 1 and it's limits are: $\omega_{min} = 1, \omega_{max} = 5$.

The current value of ω is used, in this framework's implementation, as the maximum number of agent period units that the agent should use to compute an action. It corresponds therefore to the available time limit.

4.6 The Experiment Program

In section 4.4 it was described, how the AutoFocusGlue controls the execution of the Agent and the Environment. It is through calls to the AutoFocusGlue that the user or another program, which we will refer to as the Experiment Program, illustrated in figure 4.16, can control and monitor the execution of the agent and environment and run sets of experiments.

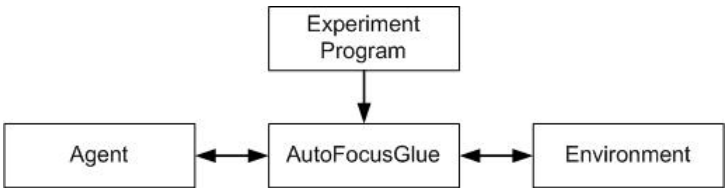

Figure 4.16: The Experiment Program in the context of the three AutoFocus subsystems.

A general Experiment Program was developed to provide an experimentation platform for the implemented AutoFocus prototypes. It provides the means to design experimentation tests and to indicate the desired outputs in a general fashion. Figure 4.17 illustrates the class model of the experiment program as the Experiment class. It is composed by an AutoFocusGlue instance, which will refer the desired Agent and Environment, one or more experimentation configurations to be executed and four instances of the AutoFocusObserver which will monitor the agent and its base mechanisms, the environment and the AutoFocusGlue. These observers can be extended to monitor specific parameters implemented in different agents or environments.

The experiments executed are declared and stored through an experimentation configuration.

4.6.1 Experiment Configuration Parameters

This section presents the complete list of the available experiment configuration parameters. All the parameters are optional unless stated otherwise.

Environment Specific Parameters

The AutoFocus platform does not impose any restrictions to the environment implementations, beyond the necessary communication interface. The environment parameters, described below, should therefore be further designed by the environment's developer.

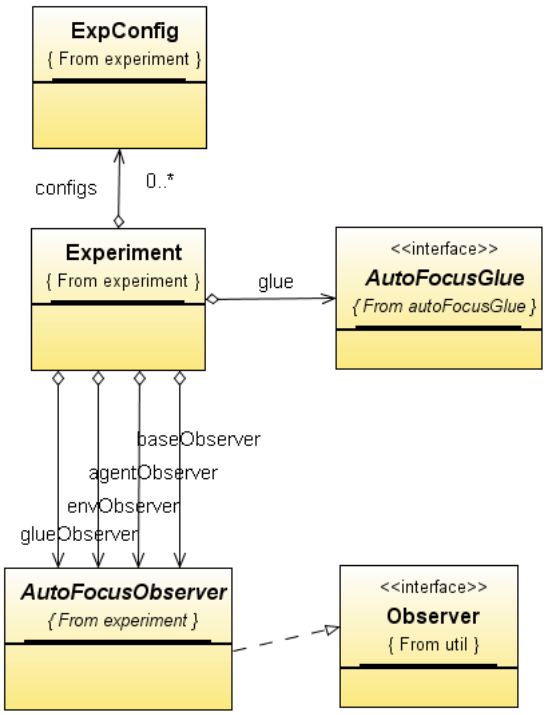

Figure 4.17: The Experiment Program class model.

- `-envInit <parameters>`
  Sets the environment initialization parameters. These parameters are set by the environment developer.
  The parameters syntax is: `param1=value1;param2=value2`.

- `-envExec <parameters>`
  Sets the environment execution parameters. They follow the same syntax as above.

- `-envMon <parameters>`
  Sets the environment monitors, i.e. indicates to the platform what are the implemented variables that the user wants to monitor.

  There is only one monitor implemented by the platform, the environment step monitor (`envStep`) allowing to observe how many times the environment has evolved.
  The parameters syntax is: `monitor1;monitor2;monitor3`. A special value "`all`" will activate all implemented monitors.

Agent Specific Parameters

- `-agInit <parameters>`
  Sets the agent initialization parameters.
  The parameters syntax is: `param1=value1;param2=value2`.

- `-agConfig <parameters>`
  Sets the agents base mechanisms parameters.
  The parameters syntax is: `param1=value1;param2=value2`.

  The available parameters correspond to the base mechanisms variables, presented in section 4.5.2:

  - `muS` - (μ_s), the permeability coefficient that determines the influence of the emotional cognitive potential p_s. (`muS=1` by default)
  - `muV` - (μ_v), the Permeability coefficient that determines the influence of the emotional cognitive potential p_v. (`muV=1` by default)
  - `alphaM` - (α^+), represents the sensibility of the attention barrier to the affective value λ^-. (`alphaM=0.3` by default)
  - `alphaP` - (α^-), represents the sensibility of the attention barrier to the affective value λ^+. (`alphaP=0.3` by default)
  - `betaM` - (β^-), represents the sensibility of the temporal focus to the affective value λ^-. (`betaM=0.5` by default)
  - `betaP` - (β^+), represents the sensibility of the temporal focus to the affective value λ^+. (`betaP=0.5` by default)
  - `epsilon` - (ϵ), the attention barrier initial value. (`epsilon=0` by default)
  - `epsilonMin` - (ϵ_{min}), the minimum limit of the attention barrier value. (`epsilonMin=-1` by default)
  - `epsilonMax` - (ϵ_{max}), the maximum limit of the attention barrier value. (`epsilonMax=1` by default)
  - `omega` - (ω), the temporal focus initial value. (`omega=1` by default)
  - `omegaMin` - (ω_{min}), the minimum limit of the temporal focus value. (`omegaMin=1` by default)
  - `omegaMax` - ω_{max}), the maximum limit of the temporal focus value. (`omegaMax=5` by default)
  - `emptyAttention` - Allows the agent to have an empty attention field or not, whether the parameter is set to "`true`" or "`false`". (`emptyAttention=true` by default)

- `-baseMon <parameters>` Sets the base mechanisms monitors.
  The parameters syntax is: `monitor1;monitor2;monitor3`
  The available monitors are: `alphaM`, `alphaP`, `betaM`, `betaP`, `lambdaM`, `lambdaP`, `muS`, `muV`, `omega`, `epsilon` and `attentionSize`.
  A special value "`all`" will activate all these monitors.

- `-agExec <parameters>`
  Sets the agent execution parameters.
  The parameters syntax is: `param1=value1;param2=value2`

- `-agMon <parameters>`
  Sets the agent monitors. Two monitors are implemented by the platform: the agent step (`agentStep`) and the time taken (`timeTaken`) monitors.
  The parameters syntax is: `monitor1;monitor2;monitor3`
  A special value "`all`" will activate all implemented monitors.

AutoFocusGlue Specific Parameters

- `-glueInit <parameters>`
  Sets the glue initialization parameters. which are the agent and environment activation rates, with the following syntax:
  `environment activation rate;agent activation rate`
  (`1;1` by default)

Experiment Execution Parameters

- `-output <output path>`
  Sets the results file output path. The results are sent to the standard output, if none is specified.

- `-name <name>`
  Experiment name (`experiment` by default). It's used to create the output file: (`output path/experiment`$_{n}ame.csv$)

- `-report <type>`
  Type of data output. Currently there are two types, `full` and `compact`. The `full` type reports the active monitors to output at every agent or environment step. The `compact` type only reports the active monitors to output at the end of each run.
  (`full` by default)

- `-end <end condition>`
  (Mandatory) The experiment end condition. An experiment ends when:

  - the agent or environment signal the AutoFocusGlue that the experiment should end
  - a specified number of AutoFocusGlue steps is executed
  - a certain environment state is achieved

  So the end condition can take one of three forms:

  - "`END`" which means that the experiment will run until the AutoFocusGlue sends the termination signal
  - a positive integer corresponding to the number of steps to be executed
  - an environment state

- `-run <experiment runs>`
  Sets the number of times this experiment will be repeated. (`1` by default)

A configuration example

Here is an example of an experiment configuration:

```
-startExp
-envMon all
-agMon all
-agConfig alphaP=0.2;alphaM=0.7;emptyAttention=true
-glueInit 10;2
-end 100
-run 10
-name configuration_example
-output \results\
-endExp
```

Note that an experiment configuration starts and ends with the keywords `-startExp` and `-endExp`.

4.7 Prototypes and Results

While implementing the AutoFocus platform several tests were made to ensure it's robustness and in the final stage of this project a full prototype was implemented, the Tileworld prototype.

This particular environment was chosen because it had already been used by Prof. Luís Morgado in his thesis[6] to prove the adequacy of the Agent Flow Model.

The Tileworld environment is characterized by a two-dimensional grid, in which the agent objective is to reach target positions, known as "holes". When the agent reaches a "hole" the "hole" disappears. The "holes" also appear and disappear randomly over time, in any free position of the grid. They have a gestation period and a life period that are determined by independent random distributions, whose characteristics are defined by the parameters of the simulation. The task of the agent is to visit as many holes as possible during the time of the simulation.

In this implementation, each hole is perceived by the agent as a motivator and the agent's current position is perceived as an observation and the agent only plans to visit the nearest "hole".

The attention field, produced by the attention focus mechanism, restricts the set of motivators ("holes"), that are considered for deliberation.

The attention field also controls the switch between planning and action. While the motivator that the agent as planned to achieve remains in the attention field, the agent will simply return the next step of the plan made to reach it. But if it disappears from the attention field, the agent will target the closest motivator present in the attention field and elaborate a plan to achieve it.

The temporal focus will determine the maximum length of the plans made by the agent. If a cognitive activity period ends during the planning process, the planning is stopped and the best partial plan formed until that time is used, returning the first action of that plan.

The dynamism of the environment was changed for each experiment and the results presented for a certain dynamism value are the average values for 100 runs of 2000 steps each. Figure 4.18 shows the values used to configure the agent and environment activation rate for each value of dynamism that we wanted to simulate.

Two separate variables were observed to measure the agents performance:

(i) the effectiveness of the agent, defined as the percentage ratio between the number of "holes" closed by the agent and the total number of "holes";

(ii) the average planning cost.

Dynamism	Agent Activation Period	Environment Activation Period
0,0	1000	1000
0,1	1000	794
0,2	1000	631
0,3	1000	501
0,4	1000	398
0,5	1000	316
0,6	1000	211
0,7	1000	200
0,8	1000	158
0,9	1000	126
1,0	1000	100
1,1	1000	79
1,2	1000	63
1,3	1000	50
1,4	1000	40
1,5	1000	32
1,6	1000	25
1,7	1000	20
1,8	1000	16
1,9	1000	13
2,0	1000	10

Figure 4.18: The relation between the Tileworld dynamism and the agent and environment activation rates.

Best results obtained

After several experiments, the following agent configuration proved to be one that provides the best results:

- alphaM = 0.7
- alphaP = 0.2
- betaM = 0.7
- betaP = 0.2
- emptyAttention = false

The remaining parameters kept their default values. Figure 4.19 illustrates the obtained results.

Here we can observe that, when the dynamism is low, the agent has time to reach practically all the holes, but when the dynamism increases, the agent does not have enough time to reach every hole before they disappear and its effectiveness decreases. That is also visible in the planning cost rise. Since there are more holes appearing

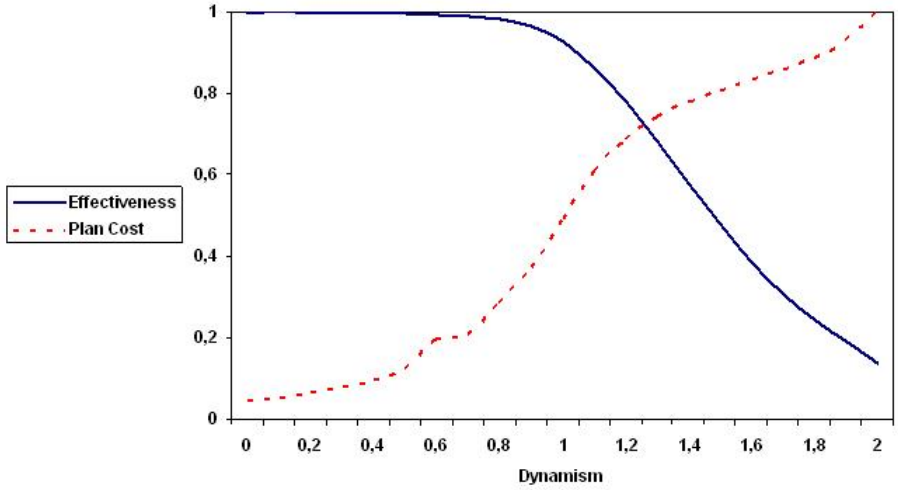

Figure 4.19: The best results obtained for the Tileworld.

and disappearing, the agent is forced to re-plan more often. The reason why the plan cost is not a linear function in comparison with the dynamism is because the temporal focus forces the agent to have shorter cognitive activity periods, meaning shorter plans.

Some additional results

Since this platform relies largely in experimentation to determine the best agent configuration, there a few more results presented below, using the previous configuration as starting point and slightly changing some of the parameters.

Allowing the attention field to be empty

- emptyAttention = true

The results, illustrated in figure 4.20, show a global decline of the effectiveness function while the planning cost remains basically the same. In this implementation, when the attention field is empty the agent takes no action which is why, even with a low dynamism the agent is incapable of reaching every hole. Possibly there could be other implementations where the agent could take a random action that would present better results.

Changing the signal of the α and β parameters

- alphaM = -0.7
- alphaP = -0.2

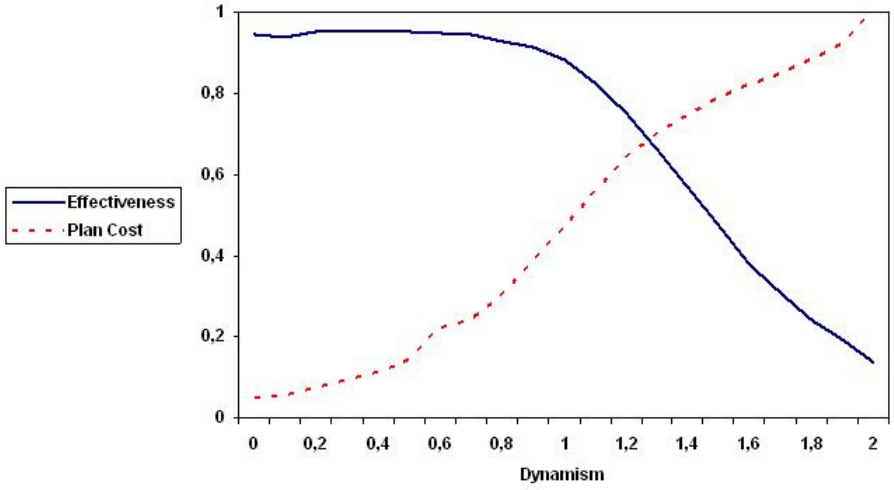

Figure 4.20: Tileworld results allowing the attention field to be empty.

- betaM = -0.7
- betaP = -0.2

By simply changing the signals of the α and β parameters the results obtained for either the effectiveness and planning cost were radically different. While the dynamism is low the agent still manages to reach almost every hole, but as soon as the dynamism rises the effectiveness drops drastically, while the planning cost rises and then drops when the dynamism reaches the highest values.

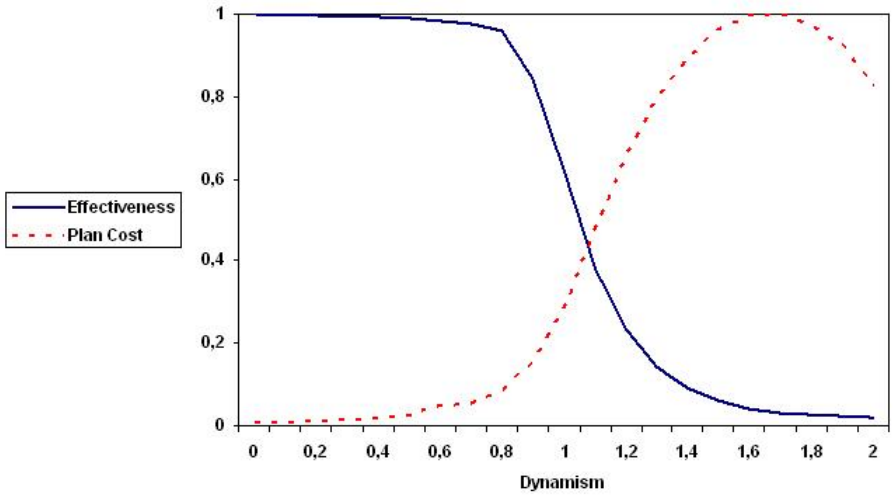

Figure 4.21: Tileworld results after changing the signal of the α and β parameters.

Using different α and β parameters

- alphaM = 0.1
- alphaP = 0.1
- betaM = 0.1
- betaP = 0.1

Using the value 0.1 for all α and β parameters gives a worse performance than the first result but it is still better than the previous ones.

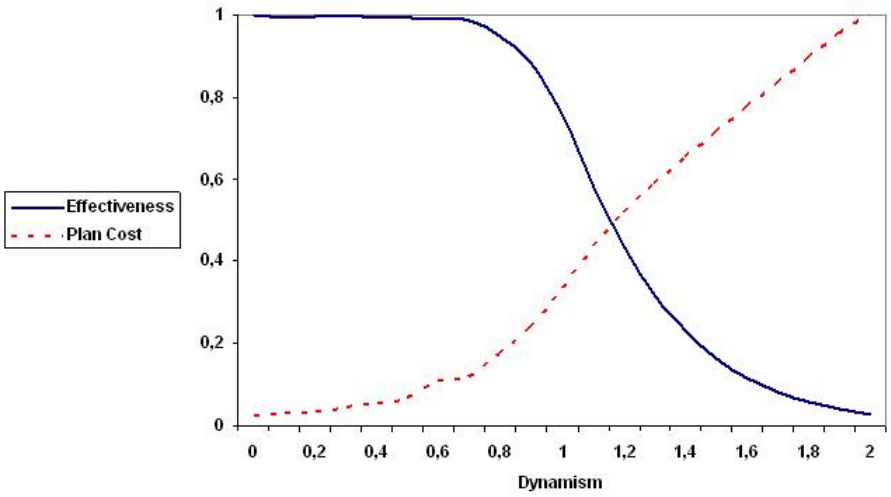

Figure 4.22: Tileworld results using different α and β parameters.

Chapter 5

Conclusions

The developed AutoFocus platform was conceived to be a flexible and general implementation of the Agent Flow Model theory. Also various aspects of the cognitive structure and base mechanisms where polished which improved the knowledge available in order to provide a general usable framework.

Although some of the original objectives were not accomplished, namely the second iteration of the platform development, the work done during this project will be the foundation upon which the memory mechanisms will be built, in the next months after the completion of this master degree. The Experiment Program was an added improvement to the initial platform requirements.

As future work, it is planned to create base mechanisms capable of defining the global emotional situation, which can be understood as the agent's emotional state, and to study how the AutoFocus agents behave in multi-agent environments.

Bibliography

[1] António Damásio. *A Second Chance for Emotion*, pages 12–22. Oxford University Press, 2000.

[2] P. Ekman and R. J. Davidson. *The nature of emotion: fundamental questions*. Oxford University Press, 1994.

[3] N. Fridja. *The Emotions*. Cambridge University Press, 1986.

[4] Peter Gärdenfors. *Conceptual Spaces*. The MIT Press, 2000.

[5] H. Maturana and F. Varela. *The Tree of Knowledge: The Biological Roots of Human Understanding*. Shambhala Publications, 1987.

[6] Luís Morgado. *Integração de Emoção e Raciocínio em Agentes Inteligentes*. PhD thesis, Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa, 2005.

[7] Luís Morgado and Graça Gaspar. Abstraction level regulation of cognitive processing through emotion-based attention mechanisms. In Lucas Paletta and Erich Rome, editors, *WAPCV*, volume 4840 of *Lecture Notes in Computer Science*, pages 59–74. Springer, 2007.

[8] Luís Morgado and Graça Gaspar. A signal based approach to artificial agent modeling. In Fernando Almeida e Costa, Luis Mateus Rocha, Ernesto Costa, Inman Harvey, and António Coutinho, editors, *ECAL*, volume 4648 of *Lecture Notes in Computer Science*, pages 1050–1059. Springer, 2007.

[9] RL-Glue. A standard for connecting reinforcement learning agents and environments. `http://rlai.cs.ualberta.ca/RLBB/top.html`.