UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MÉTHODES D'APPRENTISSAGE INSPIRÉES DE L'HUMAIN POUR UN TUTEUR COGNITIF ARTIFICIEL

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
FAGHIHI USEF

MARS 2008

UNIVERSITÉ DU QUÉBEC À MONTRÉAL

HUMAN-LIKE LEARNING METHODS
FOR AN ARTIFICIAL COGNITIVE TUTOR

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
FAGHIHI USEF

MARS 2008

I dedicate this research to Dr. Jean-Yves Housset and my parents.

ACKNOWLEDGMENTS AND SPECIAL THANKS

I wish gratefully thanks to Professor Roger Nkambou who gave me an enormous sustain to my project.

Special thanks to Dr. Daniel Dubois, CTS creator and a brilliant collaborator. I want to thank him for his patients in our discussions and all helps that he gave me to realize this project.

Special thanks to the other members of our lab who also offered me their aids, and helped me in coding phases. Mohamed Gaha, Philippe Fournier-Viger, Patrick Homeyer, thank you!

CONTENTS

LIST OF FIGURES

LIST OF TABLES

RÉSUMÉ

Les systèmes tuteurs intelligents sont considérés comme un remarquable concentré de technologies qui permettent un processus d'apprentissage. Ces systèmes sont capables de jouer le rôle d'assistants voire même de tuteur humain. Afin d'y arriver, ces systèmes ont besoin de maintenir et d'utiliser une représentation interne de l'environnement. Ainsi, ils peuvent tenir compte des évènements passés et présents ainsi que de certains aspects socioculturels. Parallèlement à l'évolution dynamique de l'environnement, un agent STI doit évoluer en modifiant ses structures et en ajoutant de nouveaux phénomènes. Cette importante capacité d'adaptation est observée dans le cas de tuteurs humains. Les humains sont capables de gérer toutes ces complexités à l'aide de l'attention et du mécanisme de conscience (Baars B. J., 1983, 1988), et (Sloman, A and Chrisley, R., 2003). Toutefois, reconstruire et implémenter des capacités humaines dans un agent artificiel est loin des possibilités actuelles de la connaissance de même que des machines les plus sophistiquées. Pour réaliser un comportement humanoïde dans une machine, ou simplement pour mieux comprendre l'adaptabilité et la souplesse humaine, nous avons à développer un mécanisme d'apprentissage proche de celui de l'homme. Ce présent travail décrit quelques concepts d'apprentissage fondamentaux implémentés dans un agent cognitif autonome, nommé CTS (Conscious Tutoring System) développé dans le GDAC (Dubois, D., 2007). Nous proposons un modèle qui étend un apprentissage conscient et inconscient afin d'accroître l'autonomie de l'agent dans un environnement changeant ainsi que d'améliorer sa finesse.

Les Mots Clé: Apprentissage, conscience, agent cognitif, *codelet*

ABSTRACT

Intelligent tutoring systems (ITS) are considered as a remarkable possibility of technology, enhanced learning which are used as assistant and even as an alternative to the human tutors. To accomplish this mission they need to maintain and use a representation of their environment that may include places, surrounding agents (artificial and human), and their actions. They can therefore take into account past and present events, and even be driven by social concerns. Since dynamic environments evolve, an ITS agent must likewise evolve to accommodate structural modifications in the environment and the addition of new phenomena. These important capacities of adaptation are observed in professional human tutors. Human being is capable to manage all mentioned complexities by the help of *attention* and *consciousness* mechanism (Baars B. J., 1983, 1988), and (Sloman, A and Chrisley, R., 2003).

However, reconstructing and implementing human capabilities in an artificial agent is far from the actual human knowledge and most sophisticated computers capacities. To achieve human-like behaviour and adaptation in machines, or simply to better understand human adaptability, we have to design human-inspired learning mechanisms. This work describes some fundamental learning mechanisms implemented in a cognitive autonomous agent, CTS (Conscious Tutoring System) developed in GDAC laboratory. (Dubois, D., 2007) We propose a model that sustains "*conscious*" and "*unconscious*" learning as a means to increase the agent's autonomy in a changing environment, and a way to improve its fitness.

Keywords: Learning, consciousness, Global Workspace theory, cognitive agent, codelet.

CHAPTER I

1. Introduction

Intelligent tutoring systems (ITS) have been used in education about since the late 1970s. Their goal is to making customized lessons or giving feedback to the learner without human involvement. In fact, ITS might maintain a separate, individual profile of each learner in order to adapt the learning session to the needs, preferences and to their current performance of the learner. Thereon, domain model (expert), learner model, tutor model and Educational Psychology (ED) represent the essential subsystems of ITS systems. Therefore, design and produce ITS systems needs meticulous caution in terms of knowledge description and possible behaviors of experts, learner and tutors to help the learner build its own knowledge in a given domain.

As information services and domains grow, domain knowledge (all information and elements of a particular domain) and reasoning mechanisms became more complicated. However, it is hard to customize and incorporate the new domains to the existing domain knowledge by a system. To challenge with this problem, Newell (Newell, A., 1990) proposed a reoriented research on modeling in psychology which capable to unify all domains. A unification of different models may use *cognitive architecture*.

Traditional robotics design was centered on having robots carry out specific tasks. Cognitive robotics adds reasoning, decision making and the ability to carry out certain additional tasks when compared to traditional robotics. A cognitive agent requires a central management organization "*mind*[1]" that includes the robot's sensory system, perception, memories, attention, Learning, action selection and effecters.

The central management organization is in charge of the overall task assigned to the agent. Cognitive agents maintain and use a representation of their environment that may include places, surrounding agents (artificial and human), and their actions. They can therefore take into account past and present events, and even be driven by social concerns. Since dynamic environments evolve, a cognitive agent must likewise evolve to accommodate structural modifications in the environment and the addition of new phenomena. To achieve human-like behaviour and adaptation in machines, or simply to better understand human adaptability, we have to design human-inspired learning mechanisms.

Our interest in this research is to add *learning capabilities to CTS*, a generic cognitive agent whose architecture is based on functional "*consciousness*" mechanisms (Baars, B. J., 1988). We aim to explain some fundamental learning methods realized in a cognitive self-governing agent, CTS. In doing so, our solution is integrated within an example ITS which aims at supporting astronauts training to use a robotic arm.

[1] **Mind (Wikipedia):** Refers to the collective aspects of intellect and consciousness which appear in certain combination of thought, perception, emotion, will and imagination.

1.1 Training the astronauts: our application platform

A robotic arm was installed on the International Space Station on April 23, 2001(Figure 1.1), during the STS-100 mission. This robotic arm, *Canadarm2*, was developed by the Canadian Space Agency and is a crucial element in assembling the station. Not only is it used to transport large loads (e.g. new modules), astronauts also use it as a mobile platform when they are working outside the station. The Arm can travel "across" the space station using the Mobile Based System[2], and from one end to the other by attaching the free end of *Canadarm* to the next available tether.

The Arm's movement can be observed by looking at the three LCD monitors of the Robotic WorkStation (RWS). The RWS has two sets of direction controls: a rotational hand controller and a translational hand controller. In addition, workstations are equipped with a control panel and a laptop computer.

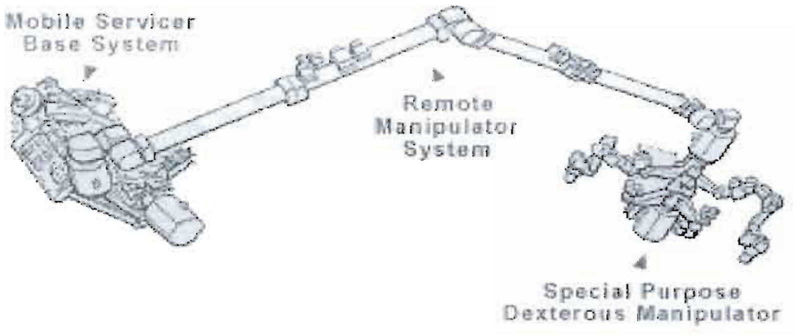


Figure 1.1 Robotic arm installed on the International Space Station

Source : http://www.nasa.gov/mission_pages/station/structure/elements/mss.html

[2] A work platform that moving along rails covering the "width" of the space station, the Mobile Base System, or MBS, provides lateral mobility for Canadarm2 as it traverses the main truss. It was added to the station during STS-111 (assembly flight UF-2) in June 2002

Manipulating the robotic arm is a difficult task, which requires astronauts to undergo a serious amount of training. The seven degrees of freedom of the arm is the first difficulty to overcome, as it considerably increases the number of possible operations. The second difficulty is sight limitation. It is impossible to have an overall view of the station; therefore, the astronaut can only see the arm through a "*steady climb*" camera installed on the station and on the Arm. Furthermore, the astronaut must choose among these cameras because there are only three screens. Figure 1.2 shows an astronaut manipulating the Arm and the three screens of *Canadarm's* workstation aboard the space station.

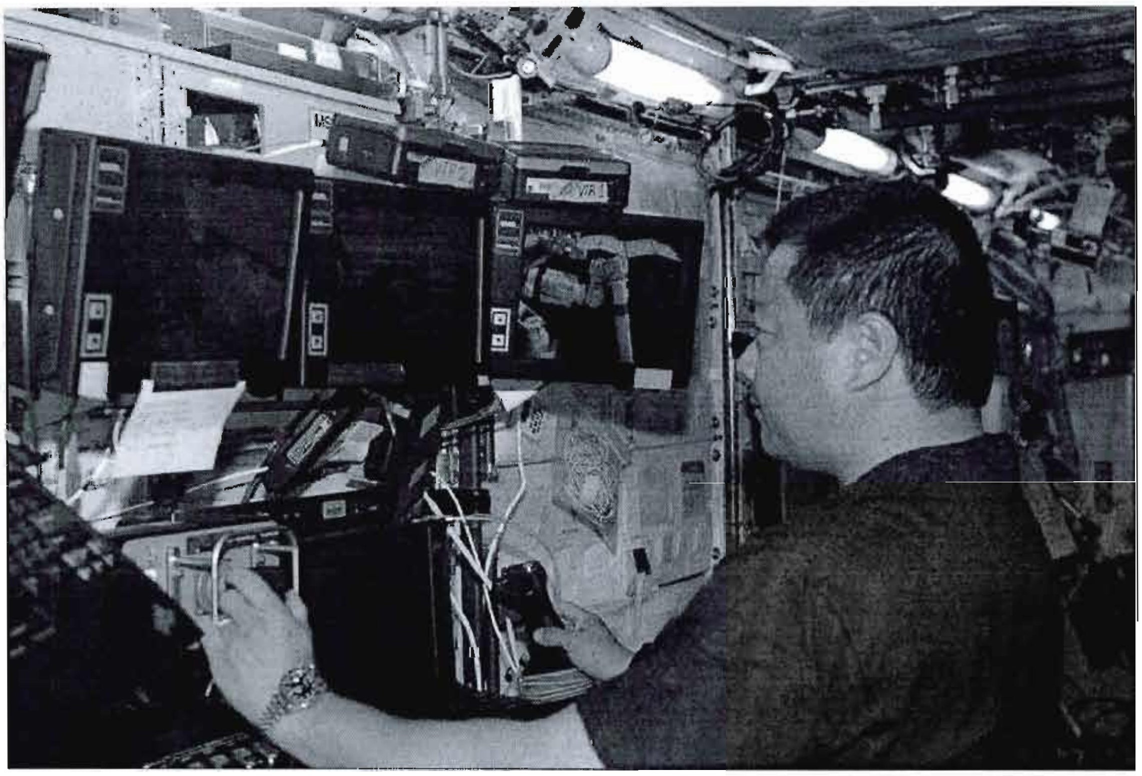
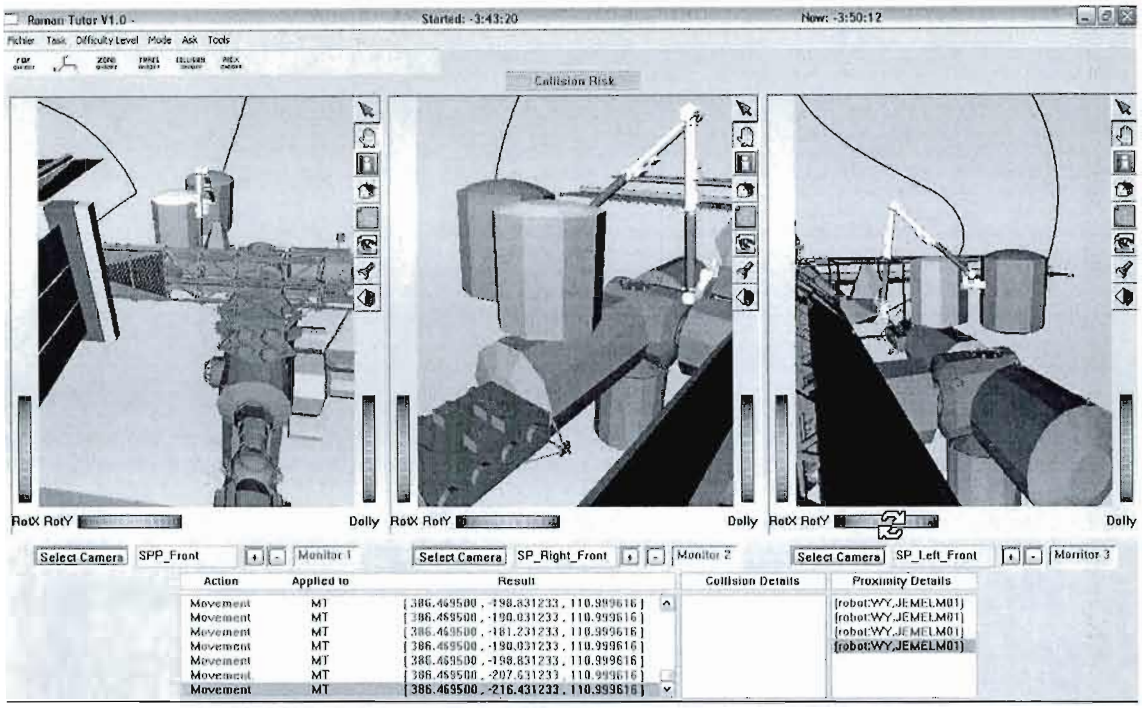


Figure 1.2 Chiao handling the Canadian Arm (Courtesy of NASA)

The astronaut must avoid moving the Arm in a way that might block it or produce a collision with the space station. Blocking is referred to as a *singularity* and constitutes a technical problem which the astronaut must consider when manipulating the Arm. Beyond the main task of manipulation comes selecting the right cameras. In addition of choosing the best views, the astronaut must readjust the cameras throughout the displacement (or movement). Our laboratory, in co-operation with the agency, developed a tutorial system, which will allow astronauts to self-learn without human supervision. Professor Nkambou (Nkambou, R., *et al.*, 2006), developed an Intelligent Training Robotic Simulator for the space station which uses an Innovative Path-Planner. The simulator makes it possible for the user to test *Canadarm* in a virtual micro-environment and to complete exercises assigned by the tutor. The role of the planner is to find a path from a given situation permitting to move *Canadarm* to the assigned destination. Astronauts are therefore provided with various contexts in which they can manipulate the Arm. The simulator executes on a standard microcomputer. Figure 1.3, shows the interface of the simulator, which reproduce the three screens that are available to the astronaut. On each screen, the astronaut can select a camera which will post images. In addition, each screen makes it possible to manipulate the viewing angles of the cameras. The interface obviously makes it possible to manipulate the arm. This is done by first selecting the joint of the arm which the astronaut wishes to move and then by swinging the angle of the joint being manipulated by using arrows of the computer keyboard. Such handling changes the state of the micro-environment. The effects can be viewed on the three screens.



Action	Applied to	Result
Movement	MT	[386.469500 , -198.831233 , 110.999616]
Movement	MT	[386.469500 , -190.031233 , 110.999616]
Movement	MT	[386.469500 , -181.231233 , 110.999616]
Movement	MT	[386.469500 , -190.031233 , 110.999616]
Movement	MT	[386.469500 , -198.831233 , 110.999616]
Movement	MT	[386.469500 , -207.631233 , 110.999616]
Movement	MT	[386.469500 , -216.431233 , 110.999616]

Collision Details	Proximity Details
	(robot:WY,JEMELM01)
	(robot:WY,JEMELM01)
	(robot:WY,JEMELM01)
	(robot:WY,JEMELM01)

Figure 1.3 A screenshot of the simulator showing a plotted path in red (Nkambou, R., *et al.*, 2006)

The astronaut can get a better understanding of what he has accomplished, and of the Arm's position, by viewing the additional information that is posted. For example, in figure 1.3, a yellow sign at the top of the middle screen indicates that there is a risk of collision. As we can see at the bottom right of the manipulations list, joint WY is approaching the JEMELM01 module. The virtual non-cognitive tutor integrated with the simulator thereby provides guidance in manipulating Canadarm2. It also helps when it traces a possible path to reach the desired final position for Canadarm2. Roman Tutor can provide a set of exercises which astronauts can select and use for practice.

1.2 Problem statement

Developing a human-like learning system with the intent of promoting knowledge to an individual is an extremely delicate problem. In our case, CTS will help astronauts to learn the manipulation of Canadarm2 in a more interactive and humanly tutoring. To do so, the agent considers a multitude of aspects: current virtual micro-world state (position and configuration of Canadarm2), potential dangers, quality of learner's actions, learner's preferences, learning objectives, and so on. Furthermore, the system must take into account the virtual world technical constraints and multi-media as well as the psychological and cultural aspects brought about by the transmission of knowledge to humans. In human beings, *consciousness* and *attention* mechanisms help manage these complexities. Such mechanisms make humans aware about the real world situation, and permit them to analyze it, to correctly resolve unexpected problems and adapt well to the unpredictable situations. That is why we considered a "*conscious*" architecture for our tutorial system.

CTS' is a "*conscious*" agent, but consciousness without learning, limits the applicability of the architecture. Some human-like learning mechanisms that CTS can be endowed with to perform more accurately are:

- Learning of Environmental Regularities: Producing algorithms which help CTS adapt to its environment rather than having a simple input-output procedure. CTS can analyze its perceptions and information that appears in its working memory; it is also capable of reviewing these concepts.

- Episodic learning: Producing algorithms which give the agent the capability of remembering and analyzing the past (events tracking) to extract relevant knowledge. It is interested in the *when*, *where* and *why* of an event.

- Procedural learning: An attempt to improve the astronaut's abilities during navigation in the virtual environment. Using the training algorithms, CTS analyzes the astronaut's mistakes and selects an exercise to help him correct them. CTS must be capable of reviewing previous errors (exercises that ended dangerously such as space collisions, poor manipulations, and so on).

We therefore can visualize that a *"conscious and learning"* architecture is a crucial base for a truly *intelligent* tutoring system. Such an agent shows behaviors that more closely parallels that of humans. In most cases, it should therefore be easier for the learner to interact with this agent. Consequently, it is easy to imagine that our architecture could be used for the development of another tutoring system.

1.3 Objective

This project clearly plans to establish learning algorithms for cognitive intelligent agents which directly interact with virtual and human environments. This elaborate work aims essentially at expanding Franklin's (Baars, B. J., 1988, 1997; Franklin, S., 2003), agent IDA (Intelligent Distribution Agent), to produce some fundamental learning methods in an artificial tutoring agent endowed with many mechanisms reproducing human consciousness and learning. IDA learns only *consciously*, but psychological experiments (cited in Faghihi,U,. *et al.*, 2007) suggest that the human being is capable of learning consciously and unconsciously. Humans may learn in many ways: by looking at something (perceptual learning), by doing a task (procedural learning) by living in or remembering an episode (episodic learning) and so on. All learning mechanisms could happen consciously and *unconsciously* (in parallel) in a human being.

In other words, our hope is to *define* and *validate* learning mechanisms inspired by human cognition as known by psychological experimentations and philosophical theories. Rather than attempting to build and provide a predetermined encyclopedic knowledge for the agent, we expressly hope to permit CTS to learn the important rules and concepts that are missing in its repertory.

1.4 Methodology

In order to achieve our goals, we initially chose an existing "*conscious*" architecture. After evaluating several architectures, we decided to base our work on the Learning Intelligent Distribution Agent (LIDA) architecture, developed by Stan Franklin at the University of Memphis. Our reason was that LIDA is a proven, universal and complete agent architecture. LIDA design and construction are based upon IDA with several forms of learning (Franklin, S., 2005). IDA was designed for US Navy personnel work (McCauley, L., and Franklin, S., 2002). *IDA modules comprise:*

> *"Perception (Zhang, Z., et al,. 1998), numerous types of memory (Anwar, A and Franklin, S., 2003; Franklin, S., et al., 2005), consciousness (Bogner, M., et al., 2000), action selection (Negatu, A., and Franklin, S., 2002), constraint satisfaction (Kelemen, A., et al., 2002), deliberation and volition (Franklin, S,. 2000). All IDA modules are based on new AI theories (Hofstadter, D. R., and Mitchell, M., 1994; Jackson, J.V., 1987; Kanerva, P., 1988; Drescher, G. L., 1991; Maes, P., 1989)."*

CTS' is a replication of IDA in a very different context: human learning.
In this sense, my final goal in this study is to extend CTS with learning capabilities in order to improve its performance. The following steps explain my work:

- Clarification of the generic notion of consciousness and logically related concepts.
- Illustration of conscious architectures and agents, in order to find patterns of how *Consciousness* could help an agent to learn.
- Identification and implementation of two fundamental types of learning (all happening consciously/unconsciously) to enrich CTS knowledge and behaviours:

  a) Learning of Environmental Regularities;

  b) Procedural learning and creativity;

Our central focus in this scientific research is on the basic principles of Perceptual and procedural learning algorithms. This portion of our work focused on the learning of the relevant concepts in the environment of the agent and the automation of complex gestures.

1.5 Document layout

First of all, I present an overview of the notion of *consciousness* and Baars' theory about *consciousness* upon which CTS' architecture is based on.

Chapter 2 presents a range of definition, architecture of agents and related domains, conscious architectures and intelligent tutorial systems.

Chapter 3 covers LIDA architecture in depth, allowing the reader to understand its functionalities.

Chapter 4 presents CTS, our "*conscious*" tutoring agent's architecture and some fundamental learning mechanisms implemented in this agent. This chapter forms the core of this document.

Chapter 5 gives a comparison between CTS (with its learning capabilities) and other popular agent architecture, as LIDA, BDI, CLARION and ACT-R.

CHAPTER II

STATE OF THE ART

2. Introduction

As with any other research project, we had a starting point, a multitude of connected works. Some of the most important aspects will be covered in this chapter. In order to assist the reader in better understanding the qualities of our architecture, the first section explains what the *consciousness* is. Then, Baars' theory of the consciousness will be clarified, the overall theory upon which our architecture is based on. We will then provide an overview of learning and of machine learning and its major theories before turning to the general meaning and architectures of agents and, its related topics which are the intelligent agents and their architectures. LIDA (Franklin, S., *et al.*, 1996-2007) architecture (developed by Stan Franklin's team at the University of Memphis), will be covered in the final section.

2.1 What is the generic notion of *consciousness*?

There are a lot of definitions offered about the concept of *consciousness* by philosophers and well informed experts which after reading and comparing them got me confused. My significant concern in this work is not to argue the validity of *consciousness*, to differentiate its characteristic, or even to find the existence of *consciousness*. I use it in a software agent for the different kind of learning, because without *consciousness,* most learning is impossible, at least in natural agents. I merely plan to show its efficacy for an artificial agent, and how it could help different learning mechanisms in CTS (Dubois, D., 2007) or any generic artificial agent.

Latest researches in cognitive sciences offer new theories, propose a wealthier, deeper sight of consciousness and its usefulness. I start by a popular example to explain the meaning of *consciousness* (after a car accident within one hour he had lost *consciousness*). A brief definition about *Consciousness* could be explained as follow:

> *"Consciousness is regarded to comprise qualities such as subjectivity, self-awareness, sentience, sapience, and the ability to perceive the relationship between oneself and one's environment. It is a subject of much research in philosophy of mind, psychology, neuroscience, and cognitive science. Some philosophers divide consciousness into* ***phenomenal consciousness****, which is subjective experience itself, and* ***access consciousness****, which refers to the universal availability of information to processing systems in the brain. Phenomenal consciousness is a state with qualia. Phenomenal consciousness is being something and access consciousness is being conscious of something." [Wikipedia]*

AI suggests that a simulation of *consciousness* can be generated in an artificial being built to resemble human cognition. Accordingly, computational mechanisms can provide an agent the majority of the same advantages that human being gifted with.

Franklin (in Franklin, S., *et al.*, 2006) suggested consciousness definition as:

> *"Conscious cognition is implemented computationally by way of a broadcast of contents from a global workspace, which receives input from the senses and from memory (Baars, B. J. 1988; Franklin, S. Baars, B. J., Rama-murthy, U., & Ventura, M., 2005; Franklin, S., 2003)."*

According to this definition to explain how *consciousness* helps learning, I give a concrete example that Dubois (in Dubois, D., 2007) offered in his thesis about a computer implementation of consciousness mechanisms. I use and change it to explain how learning methods could be integrated to the example. The CPU (the central processor unit) could be considered as the *conscious* part of a computer which processes all information coming from different parts of the system. This information could be some result from queries to a database, some mathematical computations and so on. Then the CPU causes some procedures (unconscious processes) to set-off in the computer, which prepare and return results to some active memory. There, the information becomes available for a new round of conscious (aware) processing by the CPU. The result of the unconscious processes, for instance answering the query to the database, came back to consciousness (to the attention of the CPU). These active processes may be repeated many times, thus the system should learn them, automate ("compile") them, even remember the answers in instances of stable data. Next time, the same query, does not need to be recalculated by the CPU, or at least not go through all the steps and cycles through the CPU. In this example all requests and responses might pass by CPU, the conscious part of the system which relates all information and gives them to the learning memories for learning. Although learning in current computer architectures is possible without consciousness mechanism (in my example, without intervention from the CPU), I stress here that we are dutifully explaining a human inspired architecture which mandatory must imitate human beings behavior. In fact without consciousness or cognitive mechanisms, a model may become a simple equation and its evaluation becomes a mathematical or numeric equation.

There are many methods to explain and recreate consciousness; I refer reader to Dubois' thesis (Dubois, D., 2007). However, I need to explain a specific, detailed model of *consciousness* that CTS consciousness mechanism is based upon: Baars' Global Workspace theory.

2.2 Baars' Model

Baars' model is not an imitation of *consciousness*, but rather a "*rigorous scientific theory of consciousness*" (Baars, B. J., 1988). It is a psychological theory. Our explanation is intended to allow readers to gain a better understanding of the architecture proposed and implemented by Franklin in IDA (and LIDA, its successor). I emphasize that my goal at this point is to synthesize Baars' model rather than to explain it in details or to bring a new idea about consciousness. On the other hand, we will give a metaphoric and broad view of this theory. To describe his theory, Baars uses the theatre as a metaphor.The central element of this metaphor is "*the stage of working memory*". The *scene* represents the *working memory* and the size of the scene is quite limited. In Baars' model (Baars, B. J., 1988), a large number of codelet[3] process unconscious treatments and the global workspace synchronizes and manages their activities by broadcasting their results throughout the system in order to obtain a proper conscious experience (figure 2.1).

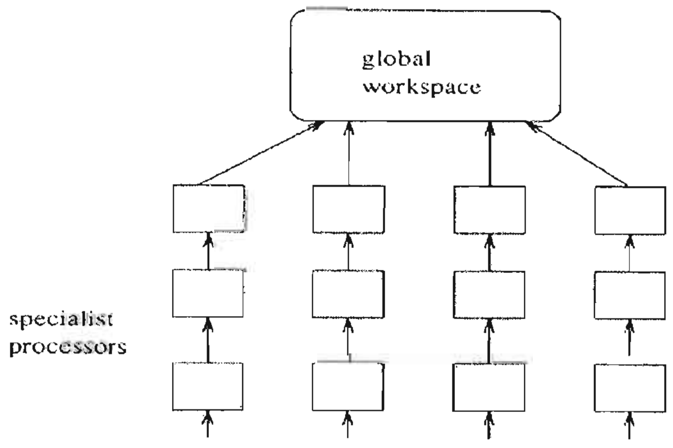


Figure 2.1 Global Workspace Barrs' Model

The scene contents are unconscious. What of it becomes conscious is what comes into the spotlight of attention, which walks through the scene. Describing the conscious part of

3 (Hofstadter, D. R., Mitchell, M., 1994), "a small piece of code executing as an independent thread that is specialized for some relatively simple task" (Negatu, A., *et al* 2006).

working memory, Baars wrote: "*The theatre has a powerful spotlight of attention, and only events in the bright spot on stage are strictly conscious.*" The actors performing on the scene compete to attract the public's attention. To illustrate the competition for consciousness on the scene, Baars' includes in his metaphorical work an example of perception. The eyes collect detected objects (when they reflect the light that is radiated towards them). It is evident that recognition is very difficult or impossible when there is insufficient light. In such context, the risk of obtaining invalid information for a given context is high. The context can include conceptual assumptions that guide our conscious thoughts and the interpretation process, even if the assumptions themselves never become conscious. An important element of the unconscious context is the *director*, representing the whole of the executive functions. The director knows the current goals of the system, and he guides the attention in working memory (and the audience's attention) according to these goals.

Finally, there is the audience, representing the unconscious processes observed in the scene and which react to events. The audience carries out most of cognitive work, unconsciously and with distributed mechanism; only the result of this work goes up on the scene to (potentially) become conscious. The structure of these actors might be quite complex; together, they create a multimodal event composed by millions of neurons, or simple (i.e. a neuron). The work of Baars is broader than this simple metaphor as he recalls scientific results which support his theory and draws upon the implications of his hypothesis.

In the next chapters I will explain how CTS could constantly learn by its needs (in fact by its attention requirements). Each time unconscious processes are chosen by attention and arrive at the conscious level, learning mechanisms start examining this information. Learning begins with a past experience (an event) or a new event.

In the next part of this chapter, I would like to describe the generic concept of *learning* and some different type of machine learning methods.

2.3 What is learning?

In 1885, Hermann Ebbinghaus described organized measurement of memory in terms of accuracy, development and capacity, as well as the impact of repetition on memory development. Actually, learning begins when an activity (for example: an action, observation, imitation, etc) leads the subject to interact with an external event. Then it could be:

- The process of gaining knowledge or skill and any change in memory (the knowledge that is stored in memory).
- The consistent remembering of an event or task.
- Understanding that we have to change our behavior through experience or conditions.
- Any stable modification in human thinking or behavior by repeating certain exercises.
- Reviewing previous experiences and trying to remember corresponding knowledge.

At the same time *learning* may be influenced by some other criteria as:

- Repetition: With repeated patterns of behavior, we are capable of predicting some features (learning the regularities).
- Importance: It is very important to remember the essence of each task that we execute.
- Order/timing: Trying to remember the when and where or causes of each event (Episodic and causal learning).
- Reinforcement: learning becomes stronger or weaker with reward or punishment

Since, I originally aim to implement different (supervised and unsupervised) kind of learning into a conscious agent; it is helpful to have a brief overview to the various types of machine learning in the next part.

2.4 Machine learning

Machine learning is a branch of artificial intelligent that focuses on algorithms and techniques that gives computers reasoning and learning capabilities.

Some types of algorithms that are organized into taxonomy:

2.4.1 Supervised learning

Supervised learning is a machine learning method for producing a function from exemplar data. The exemplar data contain pairs of input and desired outputs. As a matter of fact, machine inputs and outputs are normally present in supervised learning. The inputs can be the vectors, and the task of machine might be to predict or adjust inputs to desire outputs. Nonetheless, machines must be capable of predicting unforeseen inputs and outputs. One example of supervised learning is "*backpropagation*". However, to offer a correct answer for a given problem one needs to consider the following:

a) The type of exemplar data's,

b) Choosing cautiously exemplar data's as well as desired output for each input,

c) Choosing an apt learning algorithm for the problem at hand.

2.4.2 *Unsupervised learning*

In this learning, the training data are given to the system just as inputs, and the machine learns the correlations between these data on its own. In fact, the system has no a priori idea about outputs. On the other hand, in unsupervised learning, system gathers input data in order to use them as random variables. Unsupervised learning might be used with

Bayesian inference[4] to produce conditional probabilities. The difference between supervised and unsupervised learning is that in an unsupervised machine learning there are no input-output examples of the training data as is the case in supervised learning. The machine accepts the inputs and then attempts to adjust and translate them into a desired output on its own, without referring to the viewed? or previous examples. This learning is used for classification, pattern recognition, among others.

➢ Hebbian Learning:

Another popular methodology for unsupervised learning is Hebbian Learning (Hebb, D. O., 1949). In this kind of learning we give just input to the system and system learns how to correlate between data's and produce outputs (URL1).

" Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability.... When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased "

The role could be explained mathematically by next formula:

$$\overline{\Delta W} = \alpha \overline{X}^T \overline{Y}$$

α is the learning rate. Ergo, when an input set off an output, system might update the connection between two simultaneously active neurons which we will use this method in CTS for learning aims later on.

➢ Anti-Hebbian Novelty Filtering:

[4] Bayesian inference (Wikipedia): We use evidence or observation to infer the probability that the hypothesis may be true.

Hebbian network work properly when system receives input pattern then it set off a well-built answer. However, the answer is weak when we introduce new pattern to the system. "anti-Hebbian" learning could be produced by alpha constant negation. Some operational applications of those are:

"computer security (intrusion detection, network monitoring), machine supervision (breakdown detection), stock market supervision (detecting impeding crashes or trade irregularities), fraud detection and bio-monitoring, to name a few" (URL1).

The difference between *Hebbian* and *anti-Hebbian* learning models could be explained by the fact that the Hebbian models focuses on elements common to all the input patterns (i.e. features). However, the anti-Hebbian model focuses on each component attribute to entity patterns rather than their common features.

2.4.3 *Semi-supervised learning*

Semi-supervised learning is a combination of supervised and unsupervised learning. For example, it could be "co-training", that training strategy uses different and independent collections of examples for each learner during a course.

2.4.4 R*einforcement learning*

A discussion in reinforcement learning refers to the agent who is free to explore and react to a situation within its environment, and subsequently is able to receive positive or negative reinforcement. According to the answers (true or false) it gives to the environment, the agent will receive positive or negative reinforcement. In this method, environment could be considered as a finite-state in the *Markov decision process* (MDP). Probabilistic theory could be used in reinforcement learning to anticipate the agent's subsequent answer in MDP, to gives reward to the positive or punishment to the negative answers. A set of environments and a set of actions to be taken are present in this type of learning. A set of scalar rewards is provided each time the agent makes the best decision within its environment.

2.4.5 Case-based reasoning

Case-based reasoning (CBR) is a recent methodology came up to help the problem solving and learning. Cognitive psychological scientists believe that human being uses past experiences to solve the new problems (Aamodt, A and Plaza, E., 1994). Accordingly human uses various types of knowledge, or use a combination of various types of reasoning methods as well as representations to solve a problem and learn it. Then CBR might be capable to basically consolidate general types of knowledge in various forms. As a matter of fact, when we challenge with a new problem, we try to solve it by finding a similar past situation, and reusing its related information. CBR does not concentrate only on generalization between problems and conclusions. Actually, it tries to solve the problem by finding a similar past case, and reusing its relevant information facing within the new case. Briefly, CBR tries to find solutions about, how to utilize specific knowledge about past experiences, and then solve the new problem. It also concerns the ways in which systems can learn incrementally. As a concrete example we can talk about a physician which is diagnosing a patient. During diagnosis and treatment phase, he tries to remember and re-use the past similar symptoms treatments for the current case. In fact, Case-based reasoning looked upon as an extension of machine learning and its current challenge is finding the general reasoning techniques which could improve machine learning models. CBR second important task is offering various solutions for extracting pertinent knowledge from the skills; incorporate them as a case to the current knowledge composition and indexing relevant information. Then in this method, each time a problem is effectively solved, the know-how about the problem and offered solution might be saved to help solving alike problems in the future. When an effort to solve a problem fails, the principle causes of fail might be recognized and memorized to stay away from the same error in the future.

After giving some definitions about learning, know it is time to dutifully define the agent structure and at least how learning could be integrated to a cognitive agent.

2.5 Agents and their architectures

"*There is no universally accepted definition of the term agent.*" (Wooldridge, M., 1999).The only existing consensus is that the concept of autonomy is essential in defining an agent. However, *autonomy* is a tough concept to get detail accurately. Franklin (in Franklin, S., 2006), proposed the following definition:

"*A system embedded in, and part of, an environment, that:*

- *senses its environment*
- *acts upon it*
- *over time*
- *in pursuit of its own agenda*
- *so that its actions affect its future sensing.*"

An example would be a virus in the Windows operating system that could be activated at a precise moment.

Wooldridge (in Wooldridge, M., 1999) proposed the following definition: "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives." The key concept in this definition is *autonomous action*, meaning it acts independently, without the supervision and support from a human. In addition, an agent must learn and adapt to changes within its environment. Another characteristic of an agent is its capacity to react and recover from an unexpected event.

Our interest in a tutoring system is to develop learning mechanisms in an intelligent agent that is capable of taking *flexible* and *autonomous* actions. Such an agent should be *reactive, deliberative, hybrid and cognitive*:

- *Reactive agent*: A purely reactive agent is one whose action depends only on what it perceives in the present moment. Such an agent does not store any internal information; neither does it consider the history of its previous actions in the decision-making process. Brooks (Brooks, R. A., 1986) proposed the best known reactive agent architecture, "*subsumption architecture*". Inspired by *behavior-based robotics* in which complicated intelligent behavior is deconstructed into "*simple*" behavior modules that are organized into properly ordered layers, this architecture is based on the idea that rational behavior is the product of interaction with the environment and emerges from the interaction of simpler behaviors. That is why the agent contains a whole set of simple rules, each one reacting to a situation in the environment. These rules follow a simple form: *situation* $\rightarrow$ *action.*

  Each layer executes one of the agent's specific goals. The upper layers are more abstract. The lower layers are considered an *adaptative* part of the architecture, whereas the upper layers manage the circuit taken in executing the order to attain the *overall* objective. Decision making in the upper layer is dependent upon the lower layer. No complex reasoning must be undertaken; it is sufficient to check the preconditions of the rules. The difficulty with this model is its inability to have many layers, as the goals start to meddle with one another, resulting in low suppleness at runtime. Another problem noted by Wooldridge (Wooldridge, M., 1999) is that "*they must have sufficient information available in their local environment for them to determine an acceptable action.*" However, making decisions in an intelligent tutoring system is based on interaction with students or external environment. The tutor cannot know all there is to know about the student to reach a proper decision based on its exterior manifestations, and the environment may have hidden elements and aspects.

- *Deliberative agent* (Inman, J., Hewitt, C., 1991): These significant groups of agents are capable to monitor their environment and make an internal vision of it. They are capable to pursue their own goals by internally debating over the issues.

- *Hybrid agent* (Inman, J., Hewitt, C., 1991): It has a composite behavior of *reactive* and *deliberative agents* in that it is capable to follow its own strategy. Furthermore, it can respond immediately with some exterior events without thoughtfulness.
- *Cognitive agents*: *Newell (*Newell, A., 1990) was the first who offered the primary scheme about a cognitive agent, but he had not enough time to gradually develop his creative idea. Anderson and Lebiere (Anderson, J. R., Lebiere, C., 2003) eventually took his overall view and proposed the Newell test criteria. His criteria divided into three major categories(Hélie, S., 2007):

    - Biologic plausibility,
    - the architecture must result from an evolutionary process,
    - Learning (adaptability to its environment).

Anderson and Lebiere (Anderson, J. R., Lebiere, C., 2003) added some other criterions for a cognitive agent, such as (Hélie, S., 2007):

- the architecture must be able to operate in real time;
- the architecture must be designed in the fruitful way which supports dynamic behaviors in its reactions to the environment (to adapt its behavior according to the result of its actions on the world);
- the architecture might use a natural language(to communicate with its environment);
- meta-cognition (thinking about thinking that may be used in learning).

Newell *(*Newell, A., 1990) stated that human beings use *symbols* as their abstractions. Thus, a cognitive architecture must be able to combine the symbols ("*chunking*") in order to facilitate their subsequent uses. Then, human intelligence relies on the enormous basic knowledge. Accordingly, a cognitive architecture should be able to use its basic knowledge to form different sorts of learning (for example perceptual and procedural learning). Newell believed that the intelligence is a general function of the built up environment and a cognitive architecture must be equivalent to a universal calculation machine (Turing, A. M., 1936).

Following Newell (Anderson, J. R., Lebiere, C., 2003), and Sun (in Sun, R., 2004) proposed its own version of the cognitive architecture. Sun added some specific types of coexistent processes, as explicit/implicit processes. This terminology comes from the psychological distinction between implicit and explicit memory. The explicit process refers to the factual or declarative or non-procedural knowledge, to which meta-cognition has access. For example, the abstract idea that the moon turns around the earth. Implicit processes refer to the procedural knowledge that is in a form such that meta-cognition has no access to it. For example, how to do cycling and swimming.

In his model explicit/implicit knowledge's interact in a synergetic way to solve a problem and learn a specific task. A system using these two types of knowledge is more efficient than a system using only one or the other of these types of knowledge (Hélie, S., 2007).

However, Franklin (in Franklin, S., 2006) uses the term cognition to describe "*the endless cycle of deciding what to do next?*". LIDA, a cognitive agent based on Baars' Global Workspace theory, supports some characteristics different from the model proposed by Newell, Anderson & Lebiere and Sun. Our task is to try to understand human cognitive processes which result in the learning processes, and then implement them at a computational level.

Now, let's give more explanations about, BDI and cognitive agents.

2.5 BDI agents

Belief-Desire-Intention architecture (BDI),(Bratman, M. O., *et al.*, 1988) brings relief to those deficiencies. It has some philosophical basis in the theory of human practical reasoning. According to this theory, human intentions that are translated into action are constructed by beliefs and desires. The internal state of an agent which uses this architecture consists of beliefs, desires and intentions. Belief represents the agent's information status about itself and its environment, including other agents. Beliefs sometimes include illation rules and allow current beliefs to lead to the new beliefs. Desires (or goals) correspond to the multiple concurrent aims or situations that the agent would like to fulfill or produce. Examples of desires might include finding a good job, buying a house, opening the door. Intention represents the agent drive or motivation to approve a behavior.

The beginning of the execution of the plan or what the agent has decided to perform (the goal he has set its mind upon).

BDI architecture uses a cognitive cycle (see figure 2.2). All data entered through sensors are revised by the belief revision function (brf), which updates the existing beliefs of the agent with beliefs obtained from perceptions. For example, *Canadarm* might dangerously approach the station while it is being manipulated by the astronaut. The agent knows through its belief that a collision will occur if the astronaut continues his movement. The belief that there is a danger of collision is therefore added to the entire set of beliefs, replacing the one stating a safe status. This information eventually creates the further belief for the agent that the astronaut is a beginner.

The *generate options* function then creates options according to the beliefs and intention of the agent. This function has two goals: planning, and discovering opportunities. Planning makes it possible to determine how the agent can carry out its intentions in the current environment (all the while respecting the belief constraints). The function should discover new opportunities provided by the creation of new beliefs.

In our example, the function generates three new options: *"Allow astronaut to continue", "inform astronaut of danger", and "stop the astronaut and remedy the cause of the dangerous handling."*

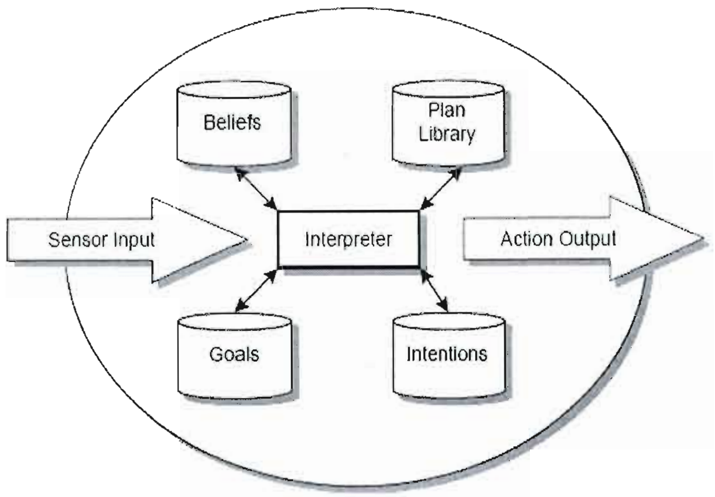


Figure 2.2 Cognitive cycle of BDI architecture. (D'Inverno, M., and Kinny, D., 1997).

The options thereby generated (desires of the agent) are then filtered by a third function (filter). This function selects the proper desire and, in the case of a multi-agent system, determines which agent will execute it.

The function is based on the current intentions of the agent as well as its beliefs in order to make this choice. In our example, the agent believes that the student is a beginner. It also believes that his dangerous behavior must immediately be remedied. That is why it chooses the final options. This choice excludes the first option and renders the second one useless (when the exercise is stopped, it is no longer useful to give a warning); hence only the third one is retained. This function also re-examines the existing intentions, removing those already carried out. It can also disengage the intention that is not helpful or realizable at the moment, given its current beliefs.

"*When does an agent disengage itself?*" That is a difficult question to answer, as both engagement and disengagement of intentions are very important. An agent that disengages too quickly may not achieve any of its intentions, whereas an agent that is not disengaged will put its energy into challenging internal intentions and soon find itself in a failing situation. In general, the agent should re-examine its intentions more often when the environment is more volatile. Finally, an *action* function chooses one of the intentions for execution that is readily achievable.

The BDI Model is an accredited model in the agent field. It is intuitive, because it encompasses human notions of reasoning. In addition, much effort was put into producing a formal model for this architecture. *The main problem in BDI architecture is that it dismisses the influence of emotions on desires and decision-making* (Vidal, H. J., and José, M., 2006).

2.6 Architecture of cognitive agent

Cognitive architecture is a design for intelligent agents which exceeds the capabilities of mentioned (for example reactive, BDI, etc) architectures. Nowadays, cognitive sciences are integrated into several disciplines related to natural cognition (see figure 2.3) such as philosophy, psychology, linguistics and neurosciences, as well as to artificial cognition (artificial intelligence and data processing). In fact, cognitive agents' architectures, in general, have different characteristics, as we mentioned above they could process information by "*symbolic computation*" in which they manipulate symbols, making tasks possible and data easily alterable by substituting one or more rules. These agents sense their environment to create information for their own use to spawn actions. Such agents might be equipped with *short and long-term memory, imagination, prediction, conceptualization, reasoning, scheduling, dilemma solving, learning, creativity, and so on* (Franklin, S., 1995).

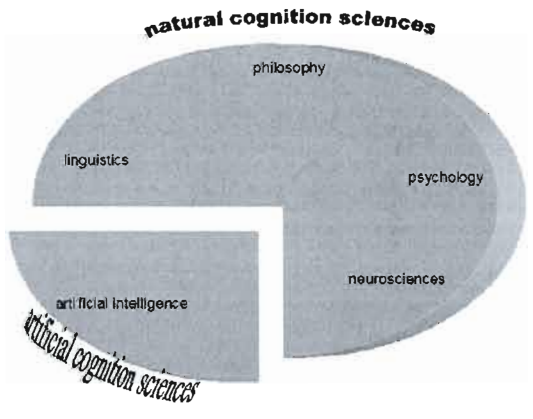


Figure 2.3 Source: Encyclopedia Universalis, 2001

Davis (Davis, D. N., 2002) proposed an architecture for a cognitive agent by focusing on "*control state, emotion, motivation and autonomy*". His view is that an autonomous agent architecture should support *drives* and *motivation*. We have to mention in this stage that, there are no universal definitions for the terms: *motivation, drive, goal and emotion*.

CogAff (Sloman, A., 2001) proposes two principal axes for the *cognitive architecture*:

I. *Cognitive cycles which includes: perception, central processing,* and *action*.

II. *Structural part:*

- Reactive mechanisms, which are (occasionally ballistic) behavioral responses to environmental and internal states;Deliberative reasoning, which requires the explicit manipulation of some form of cognitive structure (i.e. motivational constructs);
- Meta-management mechanisms (reflective processes), which monitor the agent's internal state, processes and its ongoing stimulus-response arcs.

This section illustrates Franklin's (Franklin, S., 1995) ideas about *design principles*, which have been resulting from academic analysis of many autonomous agents. It provides an overall theory of cognitive agent design.

2.6.1 Design principles for cognitive

Drives: Primary motivators for agent actions and behaviors (Davis, D. N., 2002). Drives are low-level, ecological, physiological and typically pre-conscious.

Attention: Agents have several senses; the attention mechanism helps filter unneeded inputs and focus on relevant input.

Internal models: When required, the agent builds model from its environment.

Coordination: A multi-agent system is a distributed system in that all of its components communicate to carry out a special task. There is coordination between the various parts of the multi-agent system.

Knowledge: The agent uses its senses to obtain enough information from itself and from its environment in order to execute its requirements. At the same time, acquired information might be learned (Drescher, G. L., 1988). It is evident that a better classification of knowledge and interrelations by the agent helps improve agent performance and learning. Consequently, the behavior of the agent (i.e. *decision making, reasoning*) could be closer to that of a human.

Curiosity (Franklin, S,. 1997)**:** *"If an autonomous agent is to learn in an unsupervised way, some sort of more or less random behavior must be built in. Curiosity serves this function in humans, and apparently in mammals. Autonomous agents typically learn mostly by internal reinforcement. (The notion of the environment providing reinforcement is misguided.) Actions in a particular context whose results move the agent closer to satisfying some drive tend to be reinforced, that is made more likely to be chosen again in that context. Actions whose results move the agent further from satisfying a drive tend to be made less likely to be chosen again. In human and many animals, the mechanisms of this reinforcement include pleasure and pain. Every form of reinforcement learning must rely on some mechanism. Random activity is useful when not solution to the current contextual problem is known, and to allow the possibility of improving a known solution. (This principle doesn't apply to observational only forms of learning such as that employed in memory based reasoning (Maes, P,. 1989)".*

Routines: Most cognitive agents will need some means of transforming frequently used sequences of actions into something reactive so that they run faster. Cognitive scientists talk of becoming habituated. Computer scientists like the compiling metaphor. One example is Jackson's concept demons (Jackson, J.V., 1987; Franklin, S., 1995). Agre's dissertation is concerned with human routines (in press). (Franklin, S., 1995).

2.6.2 High-level Architectures for Cognitive Agents

There is not a consensus among the architectures for cognitive agents proposed by many authors: as Albus (Albus, J. S., 1991), Baars (Baars, B. J., 1988), Ferguson (Ferguson, I. A., 1995), Hayes-Roth (Hayes-Roth, B., 1995), Jackson (Jackson, J. V., 1987), Johnson and Scanlon (Johnson, M., Scanlon, R., 1987), Sloman (Sloman, A., 1995). Franklin (Franklin, S., 1995) offers an architecture (see figure 2.4) that puts emphasis on "*action selection paradigm of mind*". The perception mechanism acquires information and gives the premier meaning to the received information. In addition, the architecture is endowed with "*long-term memory and short-term memory (workspace)*". The attention task is of choosing and filtering more relevant and urgent contexts that are related to the drives. Internal models (demonstrated) such as planners or schedulers that play a causal role in agent architecture help the agent make deliberative actions (Franklin, S., 1995). Usual actions are represented by reactions and routines; however, solving a problem requires a deliberative mechanism. Learning and reactive behaviors are the result of knowledge through which an agent can react faster or reinforce (by reasoning) the value of the actions that it has taken by reviewing its memory of past events. Learning might be connected to everything in this architecture. The lack of goals, emotions or attitudes that can influence the selection of actions in this architecture is evident. We will cover these aspects in greater depth when presenting the LIDA architecture.

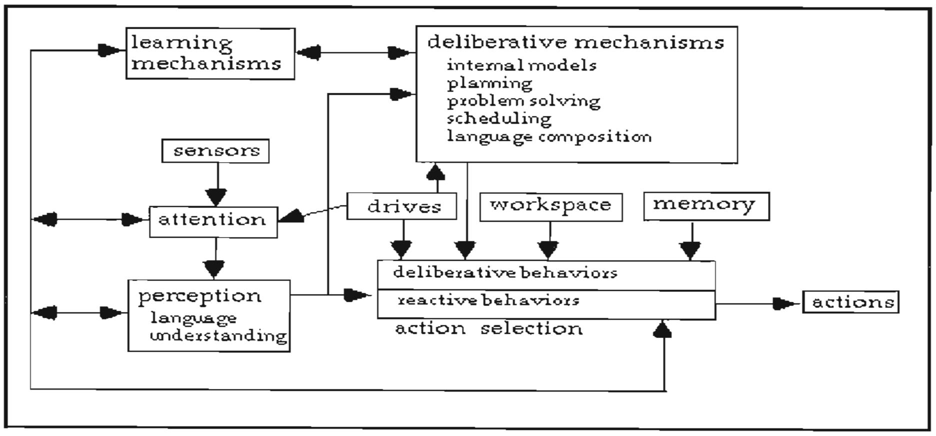


Figure 2.4 Cognitive agent architecture (Franklin, S., 2002).

In the next chapter we will see how consciousness can help an agent to learn. As a real example of a cognitive agent, LIDA uses consciousness mechanisms to learn.

CHAPTER III

BRIEF DISCUSSION ABOUT LIDA

The IDA (Intelligent Distribution Agent) architecture is the product of the in progress study of the Conscious Software Research Group at the University of Memphis. Inspired by the theory of Baars, the team of Stan Franklin (University of Memphis) successively developed three intelligent agents that implemented this theory more and more. This work began in 1996 with the Virtual Mattie agent, which manages the booking and reminders for conferences in the Memphis University. This agent was replaced by Conscious Mattie, whose implementation is more faithful to the theory of Baars. Finally, Franklin's research group developed the IDA agent for the American navy and continues to improve it. IDA eventually evolved into LIDA, the Learning IDA.

"*LIDA, the learning IDA will add three modes of learning to IDA's design: perceptual learning, episodic learning, and procedural learning. The LIDA architecture incorporates six major artificial intelligence software technologies: the copycat architecture, sparse distributed memory, pandemonium theory, the schema mechanism, the behavior net model, and the sub-sumption architecture.*" (Franklin , S., 2006).

3. LIDA: IDA endowed with learning mechanisms

The IDA task is the assignment of new billets to the sailors. In the American Navy, at the end of each sailor's tour of duty, he or she is assigned to a new billet (task) by a detailer. IDA performs the detailer's role. She communicates with sailors via E-mail and she has to understand sailor requirements and preferences, and respect all constraints (while respecting the Navy's regulations, the sailor must be satisfied by his assignment). To answer to the sailors, she has to communicate with different databases (Franklin, S., 2005). The architecture proposed by Franklin describes a multi-agent system. He called it a "*conscious agent*" since the fundamental elements and processes rely on consciousness as described by Baars'. Mainly, the agent is constructed with simple agents called "codelets" (which reproduce Baars' "*simple processors*"). The central point of the system is the "*access consciousness*", which allows all resources access to "*broadcasts*" of centrally selected information (which guide the agent to be stimulated only with the most relevant information).

LIDA's (Learning Intelligent Distribution Agent) architecture, as IDA's, is hybrid: partly symbolic and partly connectionist (according to Brooks, R. A. 1986; 1990). Human cognition facets such as perception, episodic memories, are implemented functionally, that is, with no constraint on biological validity. LIDA operates and interacts with its environment in ways inspired by human cognitive theories. Also, LIDA model allows for domain-independent perceptual and procedural learning mechanisms. This agent permits automatic learning and can repair its mistakes.

LIDA's architecture relies mostly on a "*non rules-based*" method. Here are the elements that compose it.

Workspace: The mechanism of the Workspace in LIDA corresponds to the human preconscious buffers of working memory (Figure 3.1). It is *"the manipulable scratchpad of the mind (Miyake, A., and Shah, P., 1999)"*, with decay rates of tens of seconds. Information comes to them from perceptual or other internal parts of the agent that write into the workspace. They are the "place" that hold active *codelets*, which come from perception, including previous percepts not yet decayed away, recalls from long-term memories, and structures being assembled by structure-building *codelets*. Another role of workspace is creation of links between new *codelets* (symbols) that arrive into memory and the *codelets* that already exist in the workspace (and not decayed away).

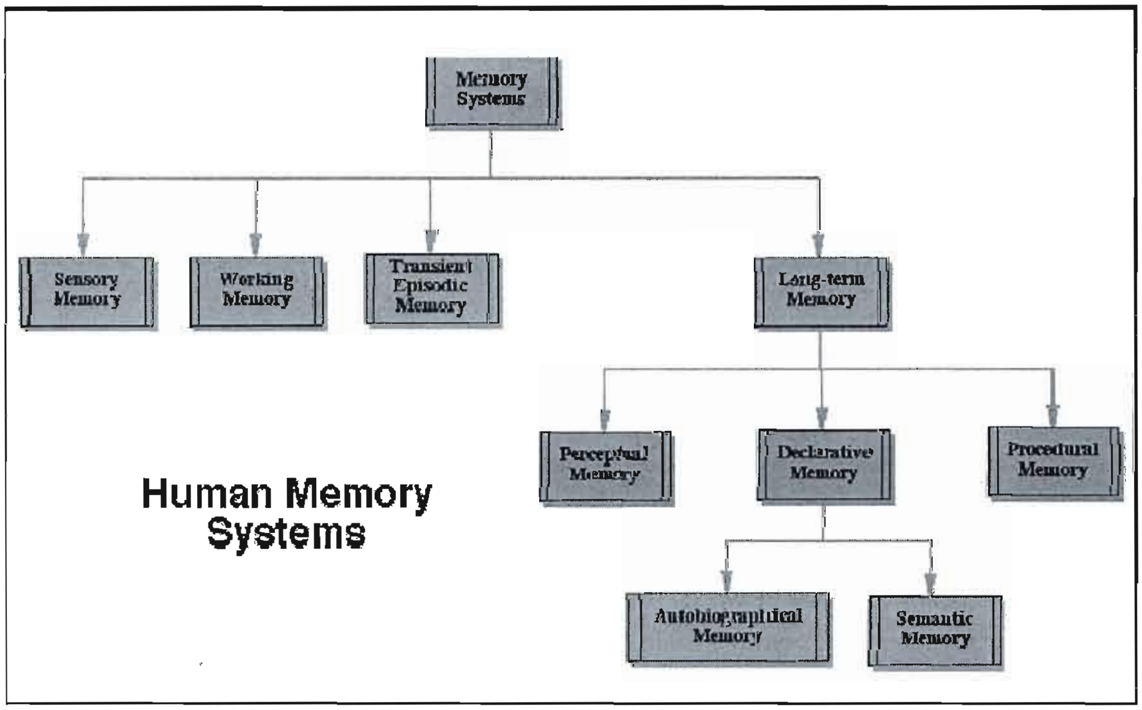


Figure 3.1 Human Memory Systems (Franklin, S., *et al.*, 2003)

Episodic memories: Memories (Figure 3.1) for events (their time, their location and what exactly occurred). LIDA has both a transient episodic memory, and a long-term, "*autobiographical*" episodic memory.

Perceptual Associative Memory: In humans, primitive feature detectors for vision are neurons located in the primary visual cortex to detect lines, etc., but for categorization and object detection, we need some association between primitive features detectors. In LIDA, it takes the form of a network of *codelets*. Perceptual *codelets* write to the Perceptual Associative memory in the form of stimulations. The *codelets,* then react and collaborate with each other to categorize the stimulus. The final pattern, a sub-network of *codelets*, then appears in the Workspace (D'Mello, S. K., *et al.*, 2006).

Functional Consciousness: In LIDA, a "*consciousness*" module implements the Global Workspace (GW) theory (Baars, B. J., 1988) in a functional way. It states that there are a lot of independent processes being in parallel processing, interacting through "*consciousness*" to accomplish a not automatized task. The functional implementation of "*consciousness*" consists of a *coalition manager*, a *spotlight controller* (corresponding to the attention), a *broadcast manager*, and *attention codelets* that recognize a new situation, or one that is interest for some reason, and interact with other *codelets* to make a common front and process a situation. What gets selected by the spotlight controller is broadcast by the "*consciousness*" to the whole system. This broadcasting is the responsibility of consciousness (D'Mello, S. K., *et al.*, 2006).

Procedural Memory: Procedural memory in LIDA is similar to Drescher's schema mechanism (Drescher, G. L., 1991). It is a graph that is constructed with nodes and links between them. Each node contains an action scheme. A scheme consists of an action, a context for it, and results that are a consequence of the execution of the scheme. Schemes can be found in the Scheme Network; they also come to exist in the Behavior Network, in their "*active form*", after their instantiation from the Scheme Net. They may be complex or simple; a simple scheme has just one action. Complex schemes contain multiple actions. Schemes need to be selected by the action selection mechanism to set off (to "fire") (D'Mello, S. K., *et al.*, 2006). LIDA uses Maes' Action Selection Mechanism (Maes, P., 1989), as modified by Negatu and Franklin (Negatu, A., and Franklin,S., 2002) to effect action selection in the service of feelings and emotions. Feeling and emotions are parallel, independent processes usually stimulated by the occurrence of events in the environment. Sometimes their urgency changes by the simple passage of time. Activation supplied by them spreads through the nodes in this direct graph. In addition, activation is coming from activation that already exists, received from predecessor actions in the Scheme Net, and from internal or external (environment) States.

3.1 Cognitive Cycle in LIDA:

LIDA's cognitive cycle (Figure 3.2) is the result of several distributed mechanisms inspired from human cognition aspects. Functional interactions are grouped in LIDA under nine sequential steps involving different parts of the agent to accomplish the processing of a perception and taking action. Cycles repeat endlessly and may be considered as beginning with a perception and ending in actions, but may be looked upon the reverse way. Baars and Franklin (Baars, B. J., and Franklin, S., 2003) suggested that cognitive cycles occurs five to ten times a second; however some processes occur in parallel and may result in much more cycles per second.

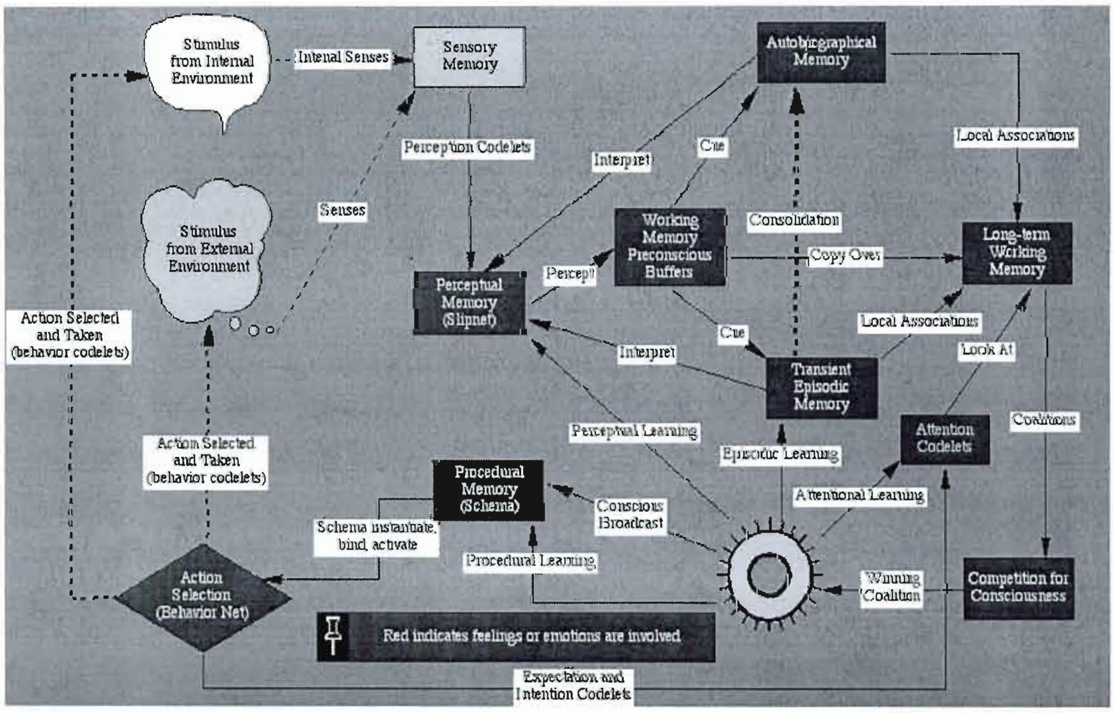

Stimulus from Internal Environment
Internal Senses
Sensory Memory
Autobiographical Memory
Local Associations
Perception Codelets
Interpret
Cue
Senses
Stimulus from External Environment
Working Memory Preconscious Buffers
Consolidation
Copy Over
Long-term Working Memory
Percept
Perceptual Memory (Slipnet)
Action Selected and Taken (behavior codelets)
Cue
Local Associations
Interpret
Look At
Transient Episodic Memory
Coalitions
Perceptual Learning
Attention Codelets
Action Selected and Taken (behavior codelets)
Episodic Learning
Procedural Memory (Schema)
Attentional Learning
Conscious Broadcast
Schema instantiate, bind, activate
Procedural Learning
Action Selection (Behavior Net)
Winning Coalition
Competition for Consciousness
Red indicates feelings or emotions are involved
Expectation and Intention Codelets

Figure 3.2 LIDA's cognitive cycle (Franklin,S., 2006)

Here are these nine steps which I give a brief description about them from Franklin and his colleagues' papers.

1-Perception:

*"The process of assigning meaning to incoming sensory data. Examples: The color red, a sound" (*Franklin, S., 2006).

2) Percept to preconscious buffer.

All interpreted data and meaning is stored in preconscious buffers of LIDA's Working Memory, adding to the information already there and that has not yet decayed away.

3) Local associations.

Preconscious buffers serve as cues for memory retrieval. Information associated with the cues are retrieved automatically from transient episodic memory and declarative memory, and stored back in Long-term Working Memory. Local associations comprise information about past actions, and feeling and emotion of the agent about these actions and events.

4) Competition for consciousness.

Here, attention *codelets* (AC) observe Long-term working memory content. AC, try to distinguish important events or urgent situations, to form coalitions describing them and bring them to consciousness. This competition is influenced by present and past feelings and emotions. Coalitions show more strength when agent affects about particular information are strong.

5) Conscious broadcast

The attention *codelet* with its related information that has been selected for its importance comes to consciousness. In humans, conscious broadcasts are hypothesized to correspond to phenomenal consciousness. Conscious broadcasts hold of all current information of consciousness and emotional part. Perceptual, episodic and procedural memories will update their information's after conscious broadcast. After each update, agent has learned and can use updated information in the next cognitive cycles.

6) Recruitment of resources.

The most relevant scheme answers to the broadcasted information by the action of its underlying *codelets*. Feelings and emotions help to find appropriate response for the broadcasted information.

7) Setting goal context hierarchy.

This answer results in the instantiation of a new goal context. Codelets request the instantiation of their corresponding node (scheme), and of the stream that contains it, including the goal. Then they bind their variables to the received information and augment their node's activation in the Behavior Network.

8) Action chosen.

The Behavior Network manager selects a scheme from current or previously instantiated behavior streams according to the presence of preconditions and on the most activated scheme. Selection of behaviors also depends on the feelings or emotions that gave activation to the *codelets* that underlie the nodes of BN.

9) Action taken.

Each selected scheme spawns at least one expectation codelet to monitor the results of the act. The execution of a behavior produces outcomes (internal or external). Expectation *codelets* observe what comes into Workspace as the result of the execution of action. They try to bring to consciousness what they find to consciousness for update or creation of new *codelets* in different part of the system (perception, episodic or procedural mechanism).

3.2 Perceptual Learning in LIDA:

One of the human learning mechanisms is perceptual learning (PL). *"Perceptual associative memory (PAM) is implemented in LIDA (D'Mello,S et al., 2006) architecture as a slipnet, a semantic net with passing activation (Hofstadter, D. R., and Mitchell,M.,1994). Perceptual learning in the LIDA model occurs with consciousness. This learning is of two forms, the strengthening or weakening of the base-level activation of existing nodes, as well as the creation of new nodes and links (D'Mello,S et al., 2006).*

Any concept that appears in a broadcast reinforces the base-level activation of the corresponding node and could reinforce its links with others in the *"Slipnet"*; this gives the node more chances to be chosen by the perceptual mechanism for future coalition. When a new object is detected by perceptual detectors, a "*new-item*" attention codelet notices this new object and tries to find common features such as spatial adjacency, common activity, and perseverance over time. If a "*similarity*" attention codelet can detect in LWM two or three items with several common features and can bring them as a coalition into "*consciousness*" for broadcasting, then a new category (node) will be created by the perceptual learning mechanism. After that, the PLM tries to find appropriate nodes to link it with. The mechanism for *Slipnet* here may look like that for an ontology. In fact, LIDA categorizes its concepts similar to the ontology mechanism to (Franklin, S., and Patterson, F. G. Jr., 2006). New relations will be learned when they are broadcast by consciousness; in LIDA, this job is done by "*relation-noting-attention-codelets*".

3.3 Episodic Learning in LIDA:

Episodic learning in human beings is the learning of the events. For example, we remember what we ate for lunch today. The first task in episodic learning is interpretation of conscious content such as What, When, Where of occurrence of each episodes into Episodic Memory.

Each time the agent finds an episode in the content of LIDA's consciousness, it tries to connect the source of activation of the current episode in the *Slipnet* to the basic features sensing elements. Each basic feature sensor in the *Slipnet* corresponds to a word, and each word has a preset place in a special buffer of the Workspace. Events happened in *consciousness* will be ciphered in Sparse Distributed Memory SDM in LIDA (D'Mello, S. K., *et al.*, 2006).

3.4 Procedural Learning in LIDA:

The authors used a combination of *instructionalist* and *selectionist* mechanisms for procedural learning in the *Scheme Net*[5]. An empty scheme consists of an *action* without *context* ,or *result*. Procedural learning happens as action reinforcement as well as the creation of new schemes. There are two kinds of activation in nodes: base-level and current-level. Base-level activation indicates the odds of its result happening if the action is taken in the proper context. Current-level, measures the pertinence of the *scheme* to the current situation. Learning (into base-level activation) goes according to a sigmoid function (a "S"-shaped curve). Initial reinforcement tends to accelerate then saturate. Here is how this learning works.

Each time LIDA executes an action, it first instantiates from the Scheme Net its empty scheme (the description of the action, devoid of context and expected result). The context (preconditions) then applied to this empty scheme is the conscious content just received before the action execution. At the same time, the *scheme* automatically produces an expectation codelet before action execution. After action execution, the expectation codelet tries to bring to *Consciousness* the results of the action. If it can, then the results are applied to the new pending node (scheme). The *Scheme Net* inserts the new node if it does not

[5] *"The scheme net is a directed graph whose nodes are (action) schemes and whose links represent the 'derived from' relation. Built-in primitive (empty) schemes directly controlling effectors are analogous to motor cell assemblies controlling muscle groups in humans. A scheme consists of an action, together with its context and its result. At the periphery of the scheme net lie empty schemes (schemes with a simple action, but no context or results), while more complex schemes consisting of actions and action sequences are discovered as one moves inwards".*(D'Mello, S. K., *et al.*, 2006).

already exist (the same action with the same preconditions and the same results; this is *instructionalist* learning). If it already exists, then the system reinforces the base-level activation of the existing node (*selectionist* learning).(D'Mello, S. K., *et al.*, 2006).

3.5 Current problems or shortcomings in LIDA:

1. When a goal is selected by the action selection mechanism, it may be routinely executed in various ways. How LIDA does make decision to execute a selected task in way1 and not in way2 or way3? To our knowledge, Franklin did not address this problem in his works. One of the current challenges (lack) in this architecture is how to select among multiple ways to realize the selected goal.

2. It is not clear how LIDA learn new *schema?*

3. As we mentioned above, the information stored in Transient Episodic Memory (TEM) represents contextual information: the nodes from PAM including some indication of physical space, plus an indication of the time of the event. This is because LIDA's TEM currently is exclusively a *perceptual* episodic memory. At this point, LIDA's research group is not sure how to encode the time of events which happened in consciousness (when an event happen) might be encoded.

4. Franklin has not yet addressed the causality of events. We believe that for each event, there is a cause. Then it would be a good thing to add a causal memory which has relations with autobiographical memory. Our proposition (Figure 3.3) is that within each coalition, it is mandatory to join a causal reason that will be used for retrieving the proper information (cognitive cycle's step 3) and learning each event.

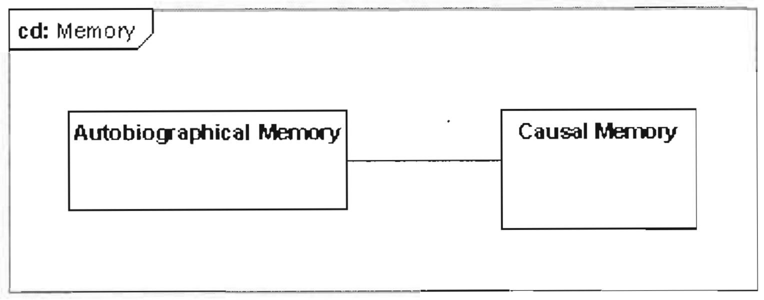


Figure 3.3 Associations between Autobiographical and Causal Memories

5. LIDA is not yet endowed with Meta-cognitive abilities. In fact, earlier attempts at adding meta-cognition to similar software agents (Zhang,Z., *et al.*, 1998; Zhang and Franklin, unpublished) was not successful. LIDA's architecture must allow to the attention codelets and "*behavior streams*" to recognize and process the need of meta-cognitive interventions.

6. LIDA currently has not addressed ascending and unconscious learning of explicit knowledge.

LIDA, in its procedural learning (D'Mello, S. K., *et al.*, 2006), proposes that the result of each action must be brought back to consciousness, whereas experiments relating to implicit learning demonstrate that satisfied expectations usually do not provide feedback to the subject (Baars, B.J., 1997; Cleeremans, A., and Jiménez, L., 1996; Cleeremans, A., *et al.*, 1998; Curran, T., and Keele, S.W., 1993).

CHAPTER IV

OUR SOLUTION: INTEGRATING LEARNING CAPABILITIES IN CTS

4. Introduction

CTS conceptual architecture is implemented as modules which communicate by messages (codelets) exchanged through Working Memory and by getting all important, urgent conversation into consciousness which inspired from the theory of Baars (Baars, B. J., Franklin, S., 2003). CTS base elements and mechanisms such as *codelets*, *consciousness*, *episodic and procedural networks, long-term memories,* are quite similar to LIDA structural design. However, there are differences in learning mechanisms, among other things. For example, *unconscious* perceptual and procedural learning were not implemented as separate mechanisms in LIDA. In fact, the existence of such a learning method in human beings is supposed in both CLARION, developed by Sun (in Sun, R., 2004) and ACT-R, developed by Anderson (Anderson, J. R., *et al.*, 2003). These systems are endowed with separate implicit and explicit learning mechanisms. However, our learning models differ from CLARION's and ACT-R's in that we used Baars' Global Workspace theory (a neuro-psychological theory) as their bases, along with psychological experimentations that are our other resources to take CTS *unconscious* learning mechanisms closer to human learning methods. Although CTS does not currently implement the whole functionality of cognition, its conceptual architecture covers a wide base (just as does LIDA's) and allows for further additions. It also supports some functional associations to the physiology of the brain.

4.1 Brief introduction to CTS cognitive cycle

As our agent is a cognitive agent, then we start this section by describing CTS cognitive cycle. The base process mechanism of the CTS system is the "*codelet*". Modules communicate with one another and they contribute information to Working Memory (WM) by information *codelets*. These travel back and forth through cycles of "conscious publications" that broadcast the most important, urgent, or relevant information. CTS cognitive cycle incorporates the traditional *Perception-Reasoning-Action* phases, but with more details (quite close to LIDA's).

CTS cognitive cycle (Figure 4.1) starts with the perception phase, with messages describing the state of the micro-world entering into CTS through its sensory buffer. This partly structured information is examined by several perception codelets that try to recognize something in the stimulus. When a perception-codelet does recognize something, it stimulates its corresponding node (codelet) in the Perception Network. The result of this collective interpretation work creates a percept that enters Working Memory (WM) as a network of codelets. At this point, codelets arriving into WM make or reinforce associations with other codelets already present. In the reasoning phase, attention codelets observe WM to find specific information and try to bring it into consciousness, competing with other coalitions that formed "*naturally*" (without the intervention of attention codelets) in WM. The Attention mechanism spots the most energetic coalition in WM and submits it to the access consciousness, which broadcasts it to the whole system. With this broadcast, any subsystem or codelets in different parts of the system that recognizes the information, may react. This may create reflection loops that pursue further a line of reasoning through consecutive participations of various sub-systems in cycles of *selection-broadcast-reaction*. Finally, the system, by its Behavior Network (BN), selects an appropriate action for execution.

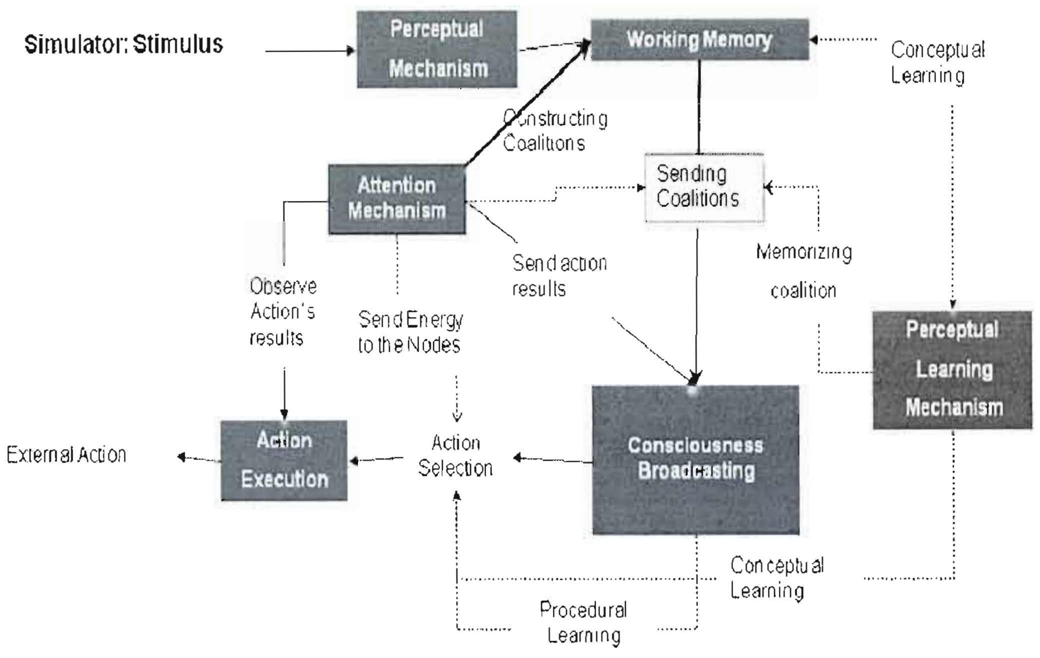


Figure 4.1 Cognitive cycle in CTS

4.2 Adding Strengths between Codelets

The first change that we made into CTS' architecture (which is useful for learning), was giving "*Strength*" to each link between two Codelets. Two codelets, will join to the WM when they have enough energy-and make connections with others, which mad links in WM could be used for learn regularities, creating elements that CTS will be able to use in later reasoning. During interaction between information codelets in Working Memory links *strength* values can be increased or decreased by a Hebbian learning (Hebb, D. O., 1949; a theory incorporated in Jackson's Pandemonium theory). According to the times codelets spend with others in WM, links between them reinforce, and then weaken while they are not together inside WM. In Figure 4.2, we can see the link between Codelet-info-Collision (by "*info*" we mean information-codelet) and two others (Joint-WE and Distance) and the strengths between them.

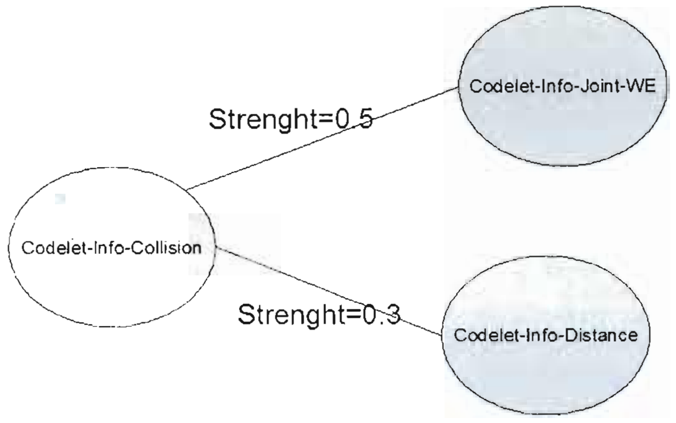


Figure 4.2 Strengths between Codelets

4.3 Access Consciousness

As we mentioned in Chapter one, for any movement in space, astronauts must first select a proper set of cameras and obtain the best possible views by adjusting them. Then only he is allowed to start moving the Arm. We will explain it by the following example. In the essay task, we suppose that the astronaut forgot to adjust its views and started to move the Canadarm. The Domain Expert module in CTS detects the problematic situation and sends a coalition describing that fact (it believes about a "Missing step: Adjusting views"). If this information gets published by the access consciousness, various modules may respond with codelets describing a probable cause, which are attached to the original coalition. Each proposition for a cause may consist of two or more parts consisting of separate codelets. In our example, one of them can arrive from the Learner Profile module proposing "Distraction" with a computed probability of 40%; a second one may be coming from the "Learner Knowledge Model" module, proposing "Poor knowledge of procedures" with a computed 60% probability. The copy of the original coalition that came back into Working Memory with the stronger cause will receive a codelet stating "Cause: Approved" from the Deliberation Arbiter. With this information, the system can make an optimal decision for any intervention, based on its available information sources. This strategy leads us to think about causal learning which I will describe with related details, later on in this chapter.

4.4 Procedural network (PN)

By Procedural network we mean planning, making decision and execution of selected actions. All mentioned parts must be mediated consciously (Franklin, S., 2006), but in CTS just actions execute consciously. The *behaviour streams* controlled and ordered under *drives* and sub-goals. In Figure 4.3 feeling or desires marked by color rings, sub-goals presented by octagons, states presented by triangles, and behaviours surrounded by rectangles. A simple example (Figure 4.3) shows "*desire*" or "*feeling*" that require to finding a remedy for collision or starting job manipulation or arbitrate, *redressing, and so on.* Behaviour selection in this architecture (Russell, S., and Norvig, P., 2003) accomplish by "*feeling*" , "*desires*" , "*state objects*" (represented as triangles), "Links between nodes" and "several thresholds".

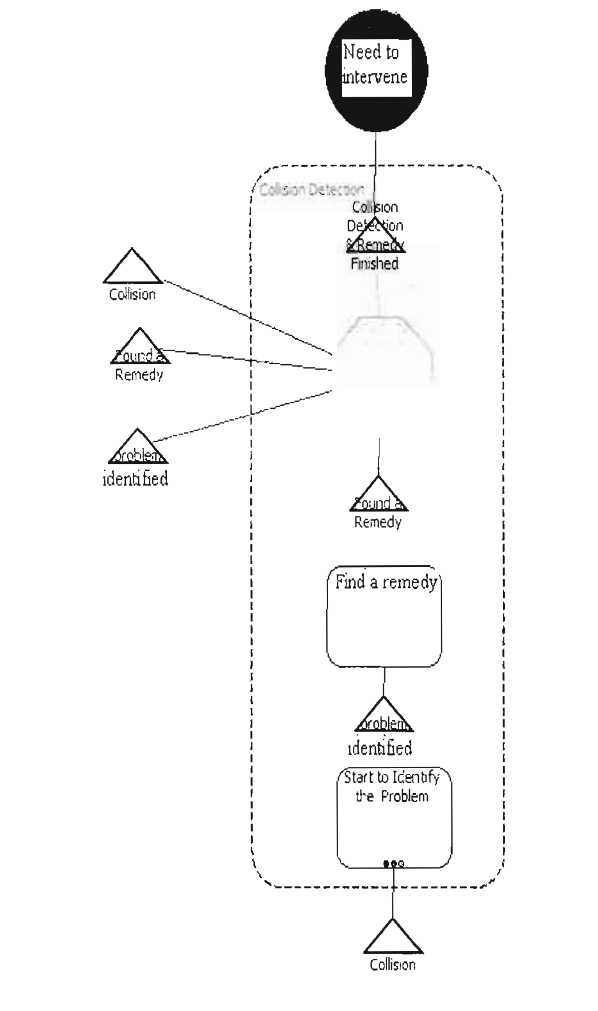
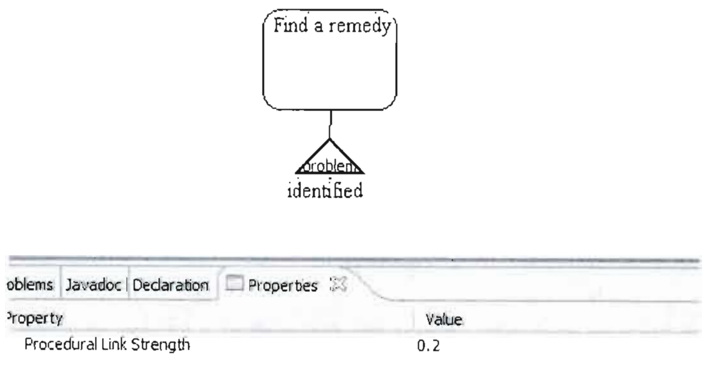


Figure 4.3 Example of a simple structure in the BN

The "*behaviour nodes*" have mandatory preconditions that will causes different sequences on the system. Received energies by "*desire*" or "*feeling*" will be spread along the system in a *top-down* model. However, "*behaviours*" may return their energies (if their preconditions change) towards other behaviours. Naturally, received energy will be spread towards next nodes, if one of the preconditions becomes true. Different scenario may be considered for a special problem. The figure 4.3 is very simple but each behaviour may have some preconditions toward others nodes and if we starting from *Collision* node, up to *Collision Detection & Remedy Finished node,* the system may pass different path to accomplish selected task (for more information we refer you to the Dubois's doctoral thesis).

4.5 Adding Strengths into Procedural Network

In the procedural network, the states listen to the publication coming from conciseness. When a state hears a publication, which contains the information that it concern about them, it reacts to the received information by changing its state to true. However, states are the preconditions of actions, "*desires*", "*feeling*" or "*behaviours*". The relations (Figure 4.4) between different nodes in the behaviour streams had not strength before, but we distinguished that this might be one of the mandatory properties for procedural learning in CTS. By giving strength to the links each time the agent select an act, their link value will change (add or lose a certain value of energies) and next time, this task will execute faster or slower (than before) according to the gathered/lost energy from previous execution.



Property	Value
Procedural Link Strength	0.2

Figure 4.4 Strength between state and other nodes in PN

4.6 CTS Learning facets

As we mentioned earlier, CTS architecture inspired from LIDA's structural design. Since our implementation differs fairly from LIDA's structural design, we propose some additional ways of learning. In this part we explain some implemented and forthcoming (which we aim to implement them in the future and are out of the capacity of a master thesis) learning aspects in CTS. Contrary to LIDA's domain of application, all inputs to the perceptual mechanism in our agent are predefined in the ISS as joints, collisions, etc. LIDA uses natural language e-mails to communicate with people and needs "*Slipnet*" to understand and giving meaning to the content of received E-mail in the Franklin(Franklin, S., *et al.*, 2006) model. CTS', does not need to discover a new world, except maybe if we later grant it with stronger conversational abilities. However, our first strategy for learning is different from LIDA learning could happens before (at perception level that we will explain it later on) and after conscious broadcast. We will discuss about Input-output learning and causal learning too.

1. The first difference between our system and LIDA is that, in LIDA, attention codelets decide to form coalition, whereas in our system the codelets create coalition by own decision and do not need attention codelets to form coalitions. For example, in our system a coalition could be a single codelet on its own, as a concrete example when user is inactive; an attention codelet (*coalition*) will be published to inform the system about inactivity in the system.

2. Learning of Environmental Regularities: LIDA accomplishes conceptual and semantic learning in its perceptual associative memory (PAM, called a *slipnet*). New concepts found in a conscious broadcast are added in the PAM with its new links. In CTS, codelets that appear within the same time-frame in Working Memory (WM) create links or reinforces their links with others. If this conjunction of codelets happens frequently, it has a high probability of corresponding to a concept and should reach a coalition status to represent it. If the repetition happens soon enough, it causes a reinforcement of links between codelets until the strength is sufficient to justify the status of coalition. Then, it may be selected for broadcasting, which allows the concept to be memorized in long-term declarative memory and in the perceptual network. We still have to design this last

step of adding concepts in the perceptual network with semantic links. Although conceptual learning may happen consciously, as in LIDA, we currently aim only at a phenomenon that happens before consciousness. I will give a more detailed description about learning of Regularities in this section later on.

3. Causal learning (underway): in this part, CTS is not limited to just input-output information. It tries to find cause and reason of each event and start to reconstruct its knowledge by them. We have not just simple I/O in CTS, agent is capable to think, and decide to form a coalition.

4. Meta cognition learning (underway): Another learning facet of the BN in CTS is the fusion of certain acts. A sort of learning that affects directly, both the planning and the action execution in BN. When a behaviour executes for many times, it is possible to create a shortcut to achieve the anticipated goal. CTS will create a new parallel pathway, eventually short-circuiting the old one. A new pathway that got created in a first execution has not enough energy for saturation, but it presents a shorter route, which is an advantage in the planning phase; if the context presents itself again many times, it will eventually saturate completely the new pathways and the old way, not used anymore, will decay away (be forgotten).

5. Procedural Learning exists in CTS: It presents an automatization capability that accelerates planning and behaviour sequences execution. This happens when energy passes through the links between acts. Contrary to LIDA, in CTS each action may refer to a decision taken by CTS learner model or domain model or a compromise between them. Then it is very important to restore this kind of information in which helps CTS, to remember (and then learn) by finding the reference of each action executed in BN.

This report presents:

Learning of Environmental Regularities, Procedural learning through reinforcement and Procedural learning through automatization.

4.7 Learning of Environmental Regularities

Cleeremans in (Cleeremans, A., *et al.,* 1996, 1998, and 2002) showed by some experiments that to perceive something consciously, it must be represented by a good quality structure that is self-sustainable for a period of time. Otherwise, it will stay unconscious. "*Perception is unconscious when it is under the subjective threshold*"(Cheesman *et al* ,1984). Dienes *et al.* (Dienes, Z., *et al*, 1997) adds to this idea that **unconscious learning may happen** when knowledge stays below the "*subjective threshold" (we added the emphasis).* So, even below *consciousness levels*, percepts may yet cause implicit learning, and this is the phenomenon we are aiming at here. In this learning, consciousness is not involved.

We reproduce this phenomenon of implicit learning of regularities by the reinforcement of the association links between codelets present in WM during the same time frame. This is inspired by the Pandemonium theory, which was developed as a bottom-up theory for categorizing; complementing this theory is the hebbian learning theory (Hebb, D. O,1949), appropriate for learning perceptual stimuli.

Pandemonium theories (Selfridge, O. G., 1959), was developed as a bottom-up theory for learning and categorizes perceptual stimuli. It consists of four layers, each layer composed of special information or demons (a function or an active micro-process) for a special task. In our agent, this learning based on the Pandemonium theory, as extended by Jackson (Jackson, J. V., 1987).

- The bottom layer consists of data and images information ("*demons"*) comes from perceptual mechanism to store and categorize concepts. Another rule in this layer is interpretation ("*Perceptual Associative Memory"*).
- The third layer composed of computational *demons*, which carry out complex computations or estimations on the data and pass results to the upper layer.
- The second layer composed of cognitive information *demons* that try to estimate the weight of data come's from third layer and try to represent appropriate information (*demons*) as candidate to the first layer.

- In the first layer, decision *demons* sees, which *demons* in second layer is more appropriate for current problem and could be broadcasted into the network.

According to Pandemonium theory, there exists population of *demons* on the arena. Arena is divided into "*stands*" and "*plaing field*". Moreover, each *demons* (we call it *codelet*) is a simple agent. Some active codelets on the *playing field* doing whatever they designed (or created) for, and the rest of the codelets situated in the *stands* are watching and waiting for something to happen in which it could excite them. After receiving enough energy for excitement, they try shouting according to the degree of stimulus that they received, if *daemon* shouts frantically enough then they can become active. When a codelet transferred from *stands* to the playing field, it starts to make relation with others. Each link has a precise weight, in a similar fashion as in a neural network.

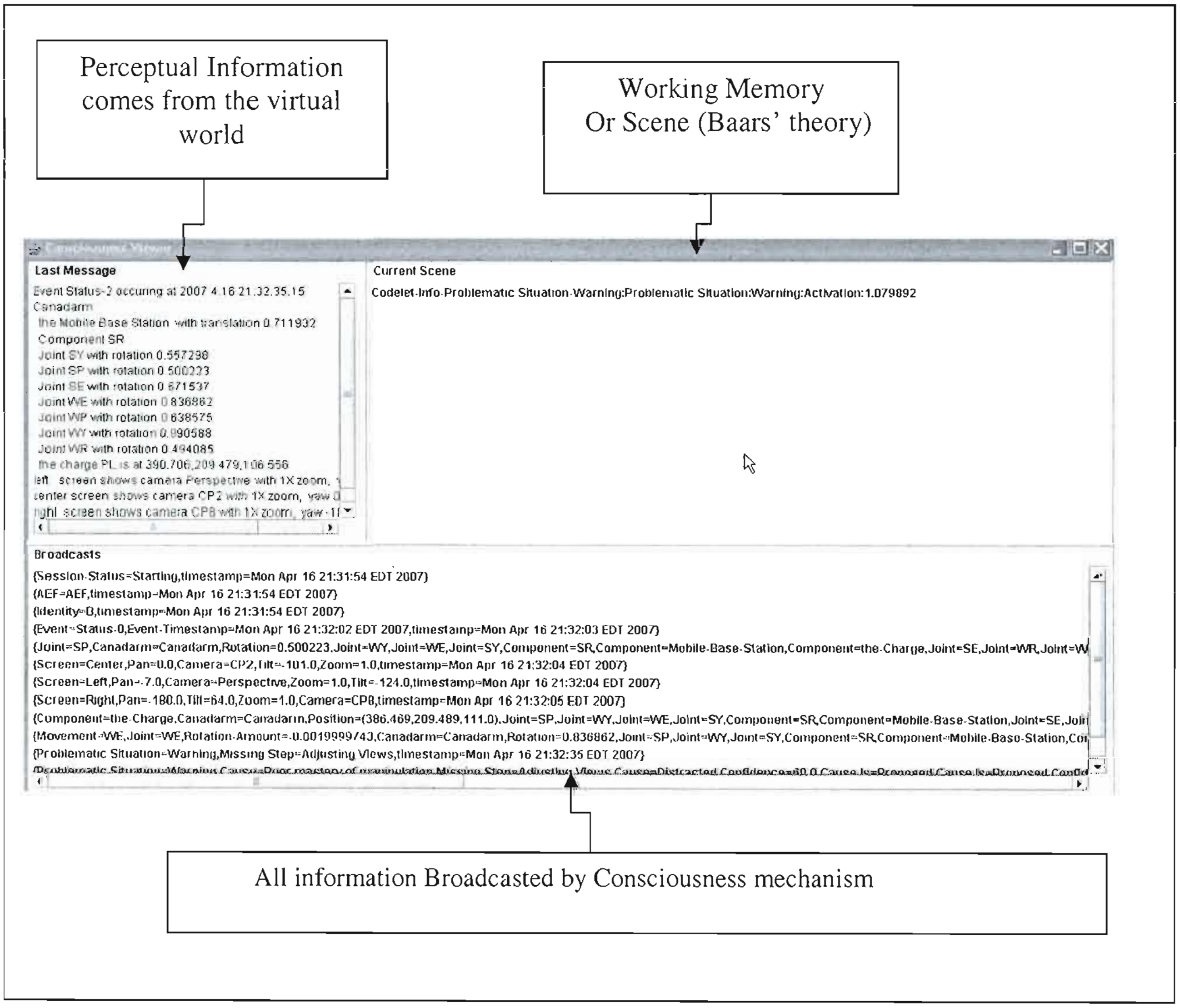
In our implementation, learning in WM is essentially related to the reinforcement of the links between codelets according to the time that they spend together in WM. This kind of learning is called *connectionist.* When a link's strength between two codelets crosses a specified threshold, these codelets learn the association as a coalition state, thereon, this new known coalition becomes available for publication by the access consciousness and can be exposed to assimilation by Working Memory. Consequently, during a cognitive cycle in CTS, virtual world information is captured by cameras then entered into WM where it will be examined by *Perceptual Learning Mechanism* (Faghihi ,U., *et al.*, 2007) (Figure 4.1).The PLM will distinguish which codelets (received information will converted to the codelets by the perceptual mechanism) has made connections with others and figures out a *meaning* (for example "*collision" status between two joints in the virtual world*) (Dubois, D., 2007). This information (codelets) with their new calculated strengths (links with others codelets in WM) will be saved as *current-active-information* (*information-codelets*), and then *information-codelets* who made a meaning and selected as *coalition* to be send to the consciousness mechanism will be examined after conscious broadcasting, to decide which of them answered by different part of the system (to produce an *action*) and might be learned. Thus, in CTS *conscious learning* happen after information's broadcasted by consciousness and *unconscious learning* (Cheesman, J., and Merikle, P. M., 1984) happen in perception phase. Both learning happen in parallel, in fact when agent doing something important at the same time perceptual mechanism receives information from virtual world and learn. Reinforcement follows a sigmoid function. Initial reinforcement happens slowly, with a subsequent rapid growth as more time of common activity is spent in WM, slowing down when the strength approaches a saturation level, until it becomes virtually permanent. Links with activation that have reached the specified maximum will weaken very slowly. Learning without any chance of forgetting would make this adaptive means a plague, maintaining any and every new link ever created. In a flesh and blood brain, energy is at premium; a connection consumes energy and must earn its keep. It must prove its usefulness. A single occurrence of a relation should not be kept unless it was created by a strong, traumatic (or highly pleasurable) event that caused a highly systemic state. In other words, the brain must be allowed to forget incidental links between neurons. Accordingly, at the end of each cognitive cycle, CTS computes the "memory loss". It scans WM to see if any link it knows of (that exists in its storage table) is used in WM (that is, the codelets using this link are present in WM). Those that are not present in WM have to incur some association loss due to the passage of time. In fact, if codelets that have links together are not called back together into WM soon enough, they will have their links weaken and eventually be forgotten by this natural phenomenon of attenuation. "Memory loss", our mechanism for forgetting, uses the same kind of sigmoid function involved in learning, but reversed and "softened" (showing a softer slope).

Indeed, we observe in real life that a concept or a procedure we learned in a few minutes can be remembered quite well for many days. For example, when learning a kata in Karate, most of the movement is remembered well enough after one or two exercises until the next training day. Here is the equation that we use for computing links' strength.

4.8 Implementation phase:

Technical implementation of mentioned descriptions in CTS appears first, when the programme executes, as a "*consciousness viewer*" (Figure 4.5), which consists of three panes:

- Last Message:
    - Perceptual information received from the simulator;
- Current Scene:
    - Working Memory (or scene, in Baars' metaphor) to which all data interpreted from the simulator and from other sources will be written temporarily.
- Broadcasted:
    - All relevant information (codelets) brought into Consciousness and broadcasted to all entities in the system;


Perceptual Information comes from the virtual world
Working Memory Or Scene (Baars' theory)
All information Broadcasted by Consciousness mechanism
Last Message
Current Scene
Broadcasts

```
Event Status-2 occuring at 2007 4.16 21.32.35.15
Canadarm
 the Mobile Base Station  with translation 0.711932
 Component SR
 Joint SY with rotation 0.557298
 Joint SP with rotation 0.500223
 Joint SE with rotation 0.671537
 Joint WE with rotation 0.836862
 Joint WP with rotation 0.638575
 Joint WY with rotation 0.990588
 Joint WR with rotation 0.494085
 the charge PL is at 390.706,209 479,106 556
left   screen shows camera Perspective with 1X zoom,
center screen shows camera CP2 with 1X zoom,  yaw 0
right  screen shows camera CP8 with 1X zoom,  yaw -1
```

```
Codelet-Info-Problematic Situation-Warning:Problematic Situation:Warning:Activation:1.079892
```

```
{Session-Status=Starting,timestamp=Mon Apr 16 21:31:54 EDT 2007}
{AEF=AEF,timestamp=Mon Apr 16 21:31:54 EDT 2007}
{Identity=B,timestamp=Mon Apr 16 21:31:54 EDT 2007}
{Event=Status-0,Event-Timestamp=Mon Apr 16 21:32:02 EDT 2007,timestamp=Mon Apr 16 21:32:03 EDT 2007}
{Joint=SP,Canadarm=Canadarm,Rotation=0.500223,Joint=WY,Joint=WE,Joint=SY,Component=SR,Component=Mobile-Base-Station,Component=the-Charge,Joint=SE,Joint=WR,Joint=W
{Screen=Center,Pan=0.0,Camera=CP2,Tilt=-101.0,Zoom=1.0,timestamp=Mon Apr 16 21:32:04 EDT 2007}
{Screen=Left,Pan=-7.0,Camera=Perspective,Zoom=1.0,Tilt=-124.0,timestamp=Mon Apr 16 21:32:04 EDT 2007}
{Screen=Right,Pan=-180.0,Tilt=64.0,Zoom=1.0,Camera=CP8,timestamp=Mon Apr 16 21:32:05 EDT 2007}
{Component=the-Charge,Canadarm=Canadarm,Position=(386.469,209.489,111.0),Joint=SP,Joint=WY,Joint=WE,Joint=SY,Component=SR,Component=Mobile-Base-Station,Joint=SE,Joi
{Movement=WE,Joint=WE,Rotation-Amount=-0.001999974J,Canadarm=Canadarm,Rotation=0.836862,Joint=SP,Joint=WY,Joint=SY,Component=SR,Component=Mobile-Base-Station,Co
{Problematic Situation=Warning,Missing Step=Adjusting Views,timestamp=Mon Apr 16 21:32:35 EDT 2007}
{Problematic Situation=Warning,Cause=Poor mastery of manipulation,Missing Step=Adjusting Views,Cause=Distracted,Confidence=60.0,Cause Is=Proposed,Cause Is=Proposed,Confid
```

Figure 4.5 CTS Consciousness Viewer

Figure 4.6, shows UML class diagrams (collaboration between different parts) of the system for learning propose.

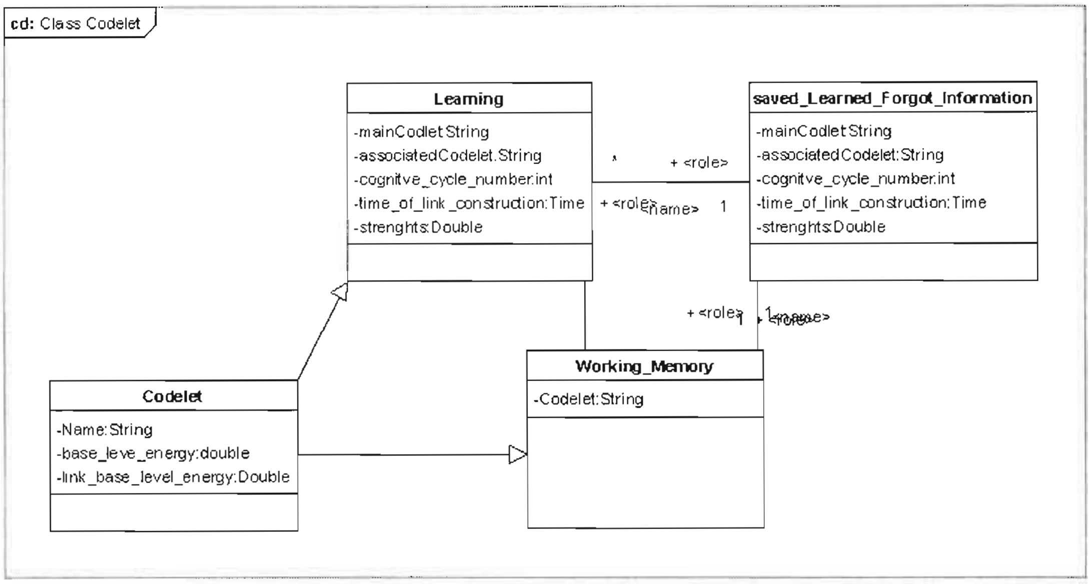

cd: Class Codelet
Learning
-mainCodlet:String
-associatedCodelet:String
-cognitve_cycle_number:int
-time_of_link_construction:Time
-strenghts:Double
saved_Learned_Forgot_Information
-mainCodlet:String
-associatedCodelet:String
-cognitve_cycle_number:int
-time_of_link_construction:Time
-strenghts:Double
+ <role>
+ <role>
<name>
1
Codelet
-Name:String
-base_leve_energy:double
-link_base_level_energy:Double
Working_Memory
-Codelet:String

Figure 4.6 Learned information in Working Memory

UML diagram (Figure 4.7) illustrates corresponding steps for learning:

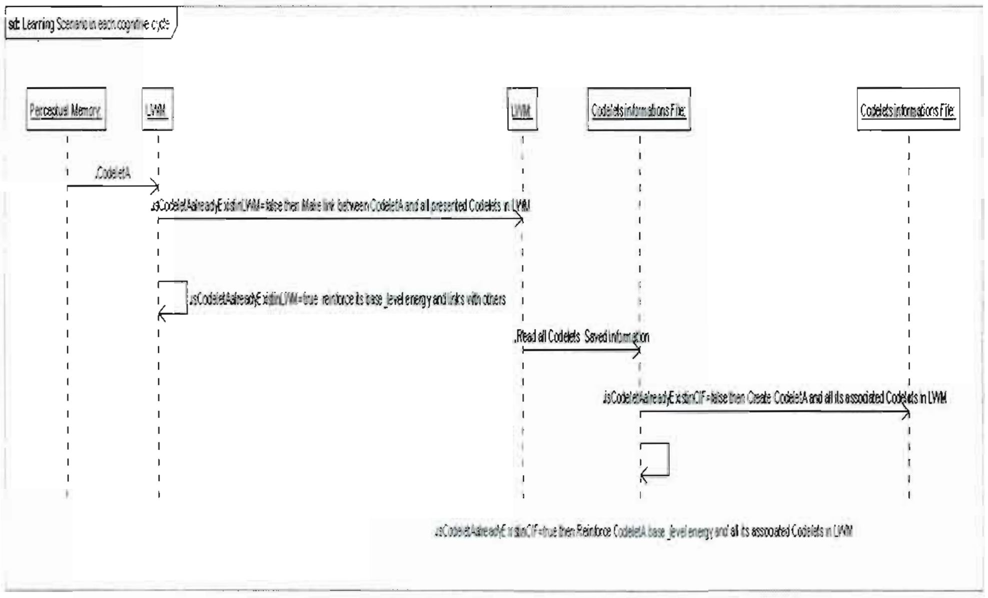

sd: Learning Scenario in each cognitive cycle
Perceptual Memory:
LWM
LWM
Codelets informations File:
Codelets informations File:
.CodeletA
.isCodeletAlreadyExistinLWM=false then Make link between CodeletA and all presented Codelets in LWM
.isCodeletAlreadyExistinLWM=true reinforce its base_level energy and links with others
.Read all Codelets Saved information
.isCodeletAlreadyExistinCIF=false then Create CodeletA and all its associated Codelets in LWM
.isCodeletAlreadyExistinCIF=true then Reinforce CodeletA base_level energy and all its assocated Codelets in LWM

Figure 4.7, Links Reinforcement and Making New Relation between Codelets

Implementation phase:

The first important task definitely given to the Learning Mechanism (LM) here is loading up all data (*information-codelets* with their links information) previously saved (*learned*) by system from storage memory (Figure 4.5), which CTS stored in previous executions.

```
private static LearningParametersSaving instance = null;

      public static LearningParametersSaving getInstance() {
            if (instance == null) {
                  instance = new LearningParametersSaving();
                instance.openFile("C:\\Learning");
            }
            return instance;
      }
```

Each codelet may have a list of links (Figure 4.8) which each links contain a strength value, with other codelets that previously saved (such as *Joint SY* with *joint SZ* in the virtual world). These links values (strength) might be changed by calling codelets into WM, and passing a portion of time with others.

```
public class LearningAssociation {

private String codelet1;
private String codelet2;
private double strength;

     }
```

All previously saved codelets with their associations will be loaded by the next procedure (getLearningAssociationsSet()) :

```
Public Set<LearningAssociation> getLearningAssociationsSet() {
            // list of associations will be find by this procedure
            Collection<List<LearningAssociation>> listes;
//          synchronized (mapAssociation) {
             listes = mapAssociation.values();


      Set<LearningAssociation> set = new
HashSet<LearningAssociation>();

            for (List<LearningAssociation> liste : listes) {
                  set.addAll(liste);
            }

            return set;
      }
```

This procedure loads up all codelets and their connections into a temporary buffer. For each cognitive cycle, we have to capture all *current-active-information-codelets* purposely launched by the software and the new ones coming up to WM. In doing so, in parallel there exist some *observer-codelets* which are aware of the system current situation; they observe and detect new events happening in the system (actOutsideScene()) and add into their list all new *information-codelets* (and their links information with others) enter WM.

```
protected void actOutsideScene() {


Set<Codelet> allActiveCodelet =Theater.getInstance().getScene()
                        .getCodelets();       }
```

New *information-codelets* which passes a portion of times and make connections with others in WM might be added to the temporary buffer. Next step is the creation of new links between new entered codelets with others (codelets) active in WM (*Scene*). Accordingly, if entered codelet from perceptual mechanism that made a *concept* has no association with others (it is new) in LM, then LM will notes all information about this new codelets with its links and other necessary information in "*allActiveCodelet*" list.

If codelet already exist in the *current-active-codelet-list of LM*, its information will be just updated which consist of the reinforcement of the links between this codelet and others (Figure 4.8).

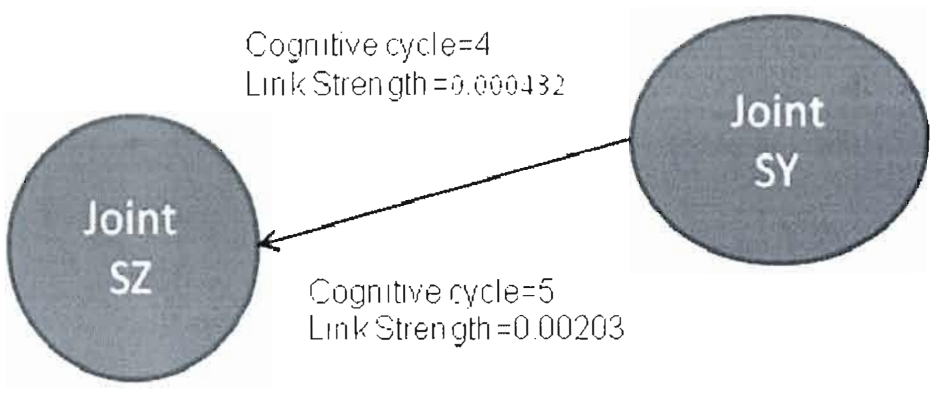


Figure 4.8 Links Strength between two codelets for two different cognitive cycles

In this stage, for each link, system has to find the last cognitive cycles in which codelet appeared and passed the time with others in WM and the last value (strength) of each links. Thus, new strength value (for each direct link between codelets) with its learning rate (here the learning rate is equal 0.1) and new cognitive cycles for each link will be computed in the sigmoid function. All these steps happend in realtime in CTS.

```
int size = oldAssociation.getTimes().size();
long dernierTemps = oldAssociation.getTimes().get(size - 1);


      int strengthSize=oldAssociation.getStrengths().size();

      oldAssociation.increaseStrength(strengthValue);
```

The saving process to ("allActiveCodelet") list happens at the end of each *cognitive cycle*. In fact, all new *codeletes* broadcasted through the system will be added automatically to the "*allActiveCodelet*" list[6] as learned *information-codelets*. At the end of the execution of software, when system receives shutdown command, then all information will be saved definitively. The reason for which we did not update entering codelets into WM, ("allActiveCodelet") list, directly into the final file ("C:\\Learning"), is that there are a lot of threads launched at the same time by the software (cognitive cycles happens five times per second in CTS), then online updating with ("C:\\Learning") will cause data interfering during data input/outputs and slow down software performance. The best way that we found was opening a temporary file which calls back all learned codelets with their links into memory, then updating this file with ("*allActiveCodelet*"), and finally, at the end of software execution, saving all information (*concepts*) by one step. Learning function follows a sigmoid function:

```
public static double sigmoidFunction(double cognitiveCycle){

double strength=1/(1+Math.pow(2.17, (rate*(cognitiveCycle)*
(System.currentTimeMillis())))));
```

Final code for saving all learned information:

```
if(currentState !=null){

      LearningStateParameterSaving.getInstance().saveToFile("C:\\Le
arning");
            }
```

[6] The information broadcasted, because they created links with other codelets in WM, and selected by attention mechanism then brought to the consciousness. This information will be learned at the end of each cognitive cycle.

Sigmoid function corresponding to our learning mechanism:

$$Strength = \frac{1}{1 + e^{(-sx+d).t.C}}$$

Where:

-x: Association strength between two codelets

-s: Rate of increase of base-level activation (for the links between codelets)

-d: Threshold value for conversion into a coalition

-C: The number of cognitive cycles since the creation of the link

-t: Time for two codelets passed together in WM.

Figure 4.9 shows the effect of the learning mechanism on the reinforcement of the links between codelets in WM, as realized with a sigmoid function. This is an example to show that learning starts reinforcing the links by small amounts. By the passage of time (with further meeting in WM), links strength tend to saturate sigmoid function very fast and then learning function will be completed. As we can see, the second half of the learning behavior in this figure follows the same behavior as the first part but inversely (saturation). By saturating sigmoid function learning rate increases very slowly (when it approach to one), however it will never reach one. The opposite of sigmoid function creates the reinforcement curve which shows the decline curve (uses for forgetting aspects). Consequently, links with low strength value incline rapidly, while links with high (saturated) strength value incline with lower rate.

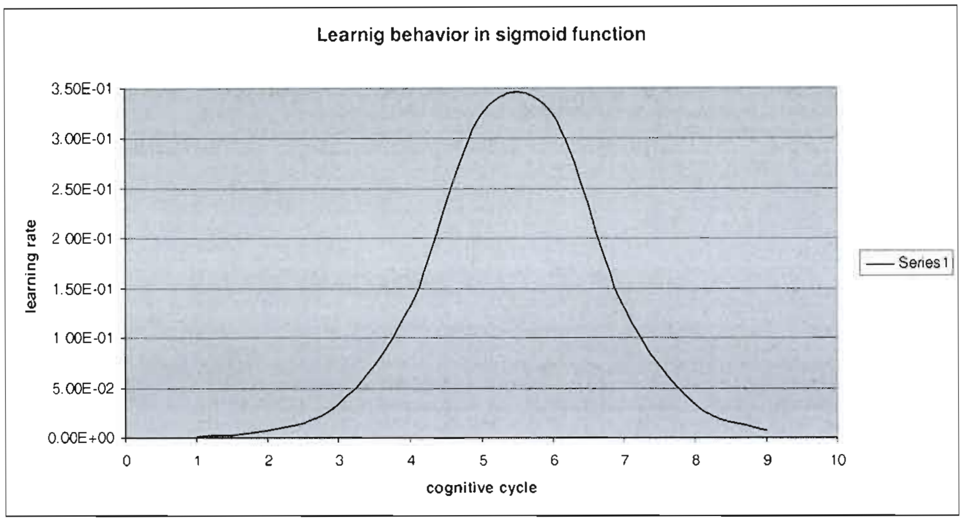


Figure 4.9, Learning rate leading to a sigmoidal curve reinforcement of the link between two information codelets.

4.9 Forgetting mechanism

Learning without any chance of forgetting would make this CTS' learning mechanism a plague, maintaining any and every new link ever created. In a flesh and blood brain, energy is at premium; a connection consumes energy and must earn its keep. It must prove its usefulness. A single occurrence of a relation should not be kept unless it was created by a strong, traumatic (or highly pleasurable) event that caused a highly systemic state. In other words, the brain must be allowed to forget incidental links between neurons. Accordingly, at the end of each cognitive cycle, CTS computes the "memory loss". It scans WM to see if any link it knows of (that exists in its storage table) is used in WM (that is, the codelets using this link are present in WM). Those that are not present in WM have to incur some association loss due to the passage of time. In fact, if codelets that have links together are not called back together into WM soon enough, they will have their links weaken and eventually be forgotten by this natural phenomenon of attenuation. "Memory loss", our mechanism for forgetting, uses the same kind of sigmoid function involved in learning, but reversed and "softened" (showing a softer slope) (Faghihi ,U., *et al.*, 2007). From the implementation point of view, here is how it goes. In each cognitive cycle, we fetch all stored active codelets (which executed in the previous cognitive cycles) and their relations with others from stored memory into a temporary buffer, and then we have to compare them with the current active codelets in WM. This, because cognitive cycles happen five times per seconds and each time there a lot of codelets enter WM(from perceptual mechanism) and system might compare new arrived codelets with previously saved(or executed by system) codelets. At the end of each cognitive cycle system might compare all codelets saved temporarily in the buffer and all active codelets in WM. If codelets which situated in temporary buffer are already present in WM, then the learning method for reinforcement will be executed for them, as described in the previous section. If system does not find them in WM, then links (strength) between codelets start to lose their energies (mean value obtained experimentally, is subtraction of current values by 0.01 which we calculate upon sigmoid function) that get weaken. Subsequently, If codelets do not get called back into WM at the end of each cognitive cycle, their links start to be forgotten by CTS forgetting mechanism, just as in human forgetting mechanism (Figure 4.10) (Hebb, D. O., 1949).

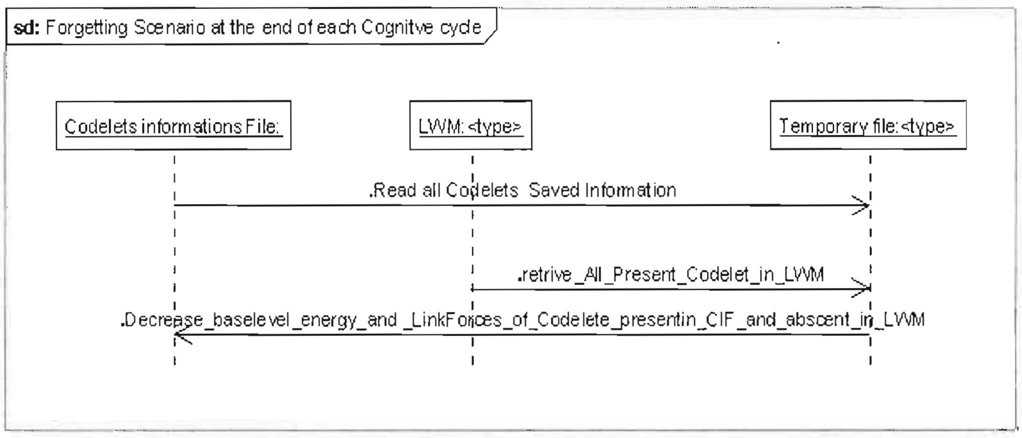

sd: Forgetting Scenario at the end of each Cognitve cycle
Codelets informations File:
LWM: <type>
Temporary file: <type>
.Read all Codelets Saved Information
.retrive_All_Present_Codelet_in_LWM
.Decrease_baselevel_energy_and _LinkForces_of_Codelete_presentin_CIF_and_abscent_in_LWM

Figure 4.10 Forgetting Scenarios in CTS

Forgetting implementation phase follows learning processes except, at the end of each cognitive cycle. Learning mechanism, try to find already learned codelets in the (*temporary-active-buffer* which loaded temporarily all learned data) memory and are not called back into Scene (software cannot find them in the *allActiveCodelet* list), then their links with other might lose energy.

```
//Find all codelet if it is active in this cognitive cycle
        boolean find=false;
        Iterator iterCodelet  = allActiveCodelet.iterator();
                        while(iterCodelet.hasNext() && find==false){
                                Codelet mine = (Codelet)iterCodelet.next();

        if(fileAsso.equalsIgnoreCase(mine.getName())){
                                        find=true;

                                }

                        }
//if codelet did not find in the WM, decrease its links //energies
with other
                        if(find == false){
                                int
sizePure=subtractionStrength.getStrengths().size();

                                double
value=subtractionStrength.getStrengths().get(sizePure-1);

        subtractionStrength.secondDecreaseStrength(value);

                        }

public void secondDecreaseStrength(double cognitiveCycle) {
                double baseValue = Math.log10((1 - cognitiveCycle) /
cognitiveCycle);
        double e = Math.log10(2.17);
        double bdivtoe = (baseValue / (-2 * e));
        double x = bdivtoe - 0.01;
        double secondStrength = 1 / (1 + Math.pow(2.17, ((x - 5))));
```

Here, updating processes and saving all data's after stopping software, follows learning methods.

To illustrate learning and forgetting, Figure 4.11 and Table 4.1 show how link strength evolves between two codelets (Codelet-Info-Event-*, Codelet-Info-Event-Timestamp-*) called into WM at the same time. Upon their appellation into WM, the current link between them has a strong value (Strength =0.9984). It means that in the past cognitive cycles, this pair of codelet has been called several times into WM. As we can see, after their meeting at the first cycle considered in the illustration, they have not been called into WM for the next

14 cycles, so they lost some of their association strength with the passage of time. With their common recall into WM at cycles 16 and 30, their link reinforced, followed with forgetting. Up After the 30th cycle, they have never been recalled into WM. In fact their relations (links between them with its energy) decreased constantly.

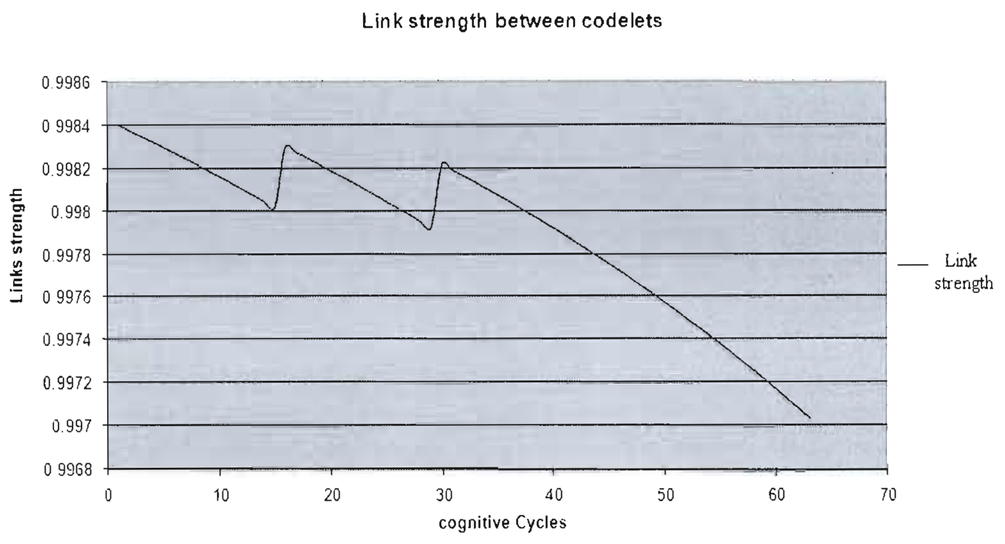


Figure 4.11 Strengths between codelets in forgetting scenario

1	2	3	4	5	6	7	8	9	10	11	12
0.998401	0.998376	0.998351	0.998273	0.998299	0.998273	0.998246	0.998218	0.998191	0.998162	0.998134	0.998105
13	14	15	16	17	18	19	20	21	22	23	24
0.998075	0.998045	0.998015	0.998299	0.998273	0.998246	0.998218	0.998191	0.998162	0.998134	0.998075	0.998045
25	26	27	28	29	30	31	32	33	34	35	36
0.998015	0.997984	0.997952	0.99792	0.998218	0.998191	0.998162	0.998134	0.998105	0.998075	0.998045	0.998015

Table 4.1 Strength values between Codelet-Info-Event-*, Codelet-Info-Event-Timestamp-*

Now we discuss about different scenarios that could happen in CTS memory. Figure 4.12 shows learning and forgetting scenarios in CTS' WM. The solid (blue dark) curve (Codelet-Info-Joint-SY and Codelet-Info-Component-SR) shows how the strength of a link grows if a pair of codelets is brought back into WM as soon as it leaves it, never being exposed to "memory loss". The link between codelets is constantly reinforced while they spend time together in WM. The dashed pink (dark) curve shows a different scenario. Learning starts and continues up to the 50th cycle, followed by a short "memory loss" (during a few cycles spent outside WM), then the two codelets meet again in WM, reinforcing their link for a brief period. After about 25 cycles outside WM where their association weakens, they see a final learning episode. You will note that forgetting follows a softer slope than for learning. With the third, yellow (bright) curve, we can observe a third scenario for the link between two codelets: it is created, grows initially, then when one codelet disappears from WM and never comes back, its association eventually vanishes.

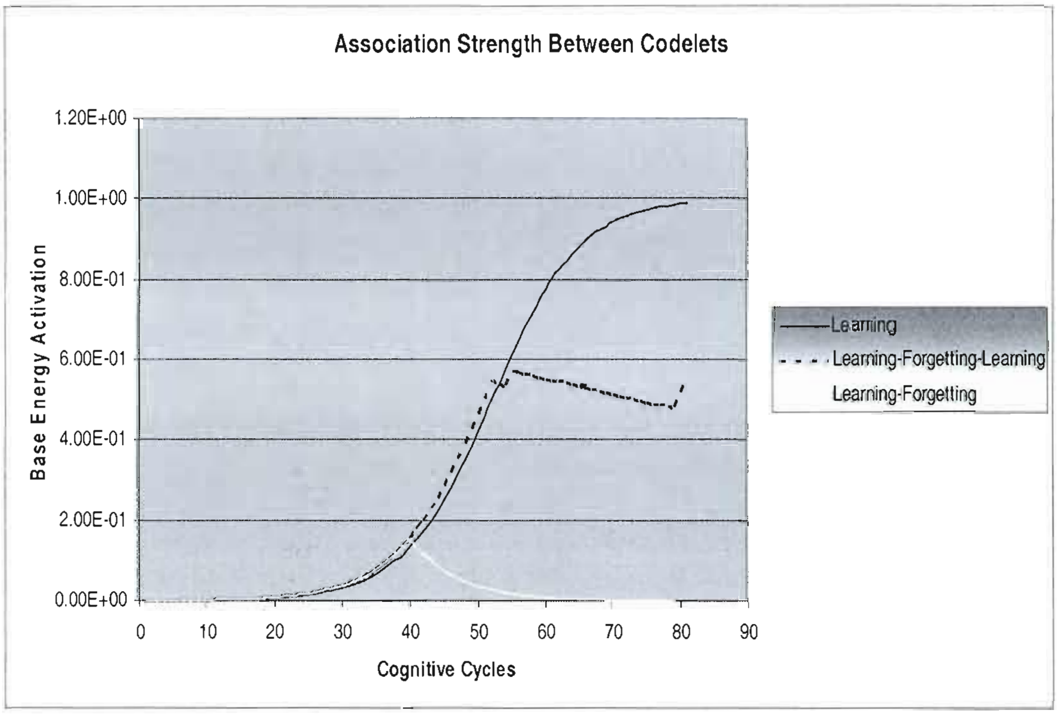


Figure 4.12 Learning and forgetting between codelets in CTS

With this mechanism, CTS uses an unsupervised machine learning strategy that accomplishes a *selectionist* adaptation of the agent. This is appropriate, as system does not know ahead of time which codelet will enter WM and which codelet will eventually form the next coalition. Computationally speaking, all information about the new computed strength is saved, to be used in the next cognitive cycle or when the agent is reactivated for a new session.In this kind of Hebbian learning[7], agent uses an unsupervised machine learning strategy. As we can observe above, link values between different codelets changes according to the cognitive cycles and the time that they passed together in WM. In the first premier steps, the increase level of learning is started by small amount then after some cognitive cycles incremented faster with bigger amount.

[7] See http://en.wikipedia.org/wiki/Hebbian_learning

4.10 Procedural learning in CTS

There are two ways for procedural learning; the first one is creating new behaviour nodes. After a scheme is chose (because, among other rules, its context for execution becomes true), an action is executed which then spawns an expectation codelet to bring to consciousness the result of the executed action. In the next step, the agent decides to create a new procedural node according to the novelty of task or reinforce the existing link activation degree if current task already exist (D'Mello, S. K., *et al.,2006*).

The other procedural learning in CTS is the *implicit* procedural learning. Each time a state (*pre-conditions* showed as triangular in Figur 4.13) becomes true, CTS learning mechanism make a copy of executed codelets (Figure 4.13., the small blue circles in the rectangular shows codelets). In this method, the links value of each node (nodes showed as the rectangular in the Figure 4.13) executed in the behavior network with its subsequence node will get reinforced. In the future executions, each time the same two codelets get fire by behavior network, links between nodes will get reinforced .The traces of the executed nodes will be saved as the learned paths.

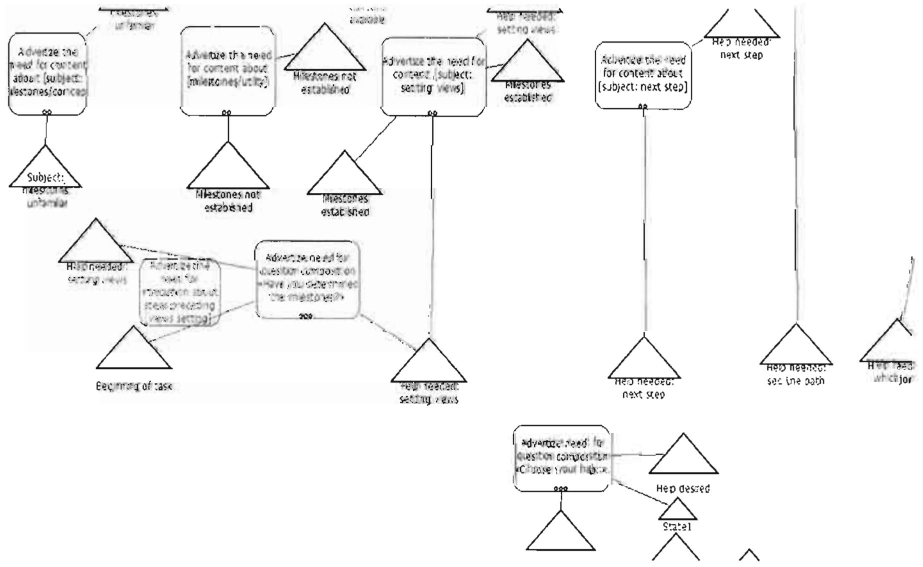

Milestones unfamiliar
Advertize the need for content about [subject: milestones/concep
Advertize the need for content about [milestones/utility]
Milestones not established
Advertize the need for content [subject: setting views]
Help needed: setting views
Milestones established
Advertize the need for content about [subject: next step]
Help needed: next step
Subject: milestones unfamiliar
Milestones not established
Milestones established
Help needed: setting views
Advertize the need for instruction about steps preceding views setting]
Advertize need for question composition «Have you determined the milestones?»
Beginning of task
Help needed: setting views
Help needed: next step
Help needed: see the path
Help need: which jor
Advertize need for question composition «Choose your help»
Help desired
State1

Figure 4.13 Part of CTS' behavior network concerned with offering support.

4.11 Implicit procedural learning in CTS

D'Mello (D'Mello, S. K., et al., 2006) proposed some procedural learning in LIDA that adds new nodes in its schema net. Here we mainly propose complementary procedural learning mechanisms that effect implicit procedural learning, "*implicit*" having two possible meanings here: that the learning happens unconsciously, and that the representation takes a sub-symbolic form (which cannot be used in conceptual manipulations). The resulting benefit of this learning on the implicit structures is faster operations (but not the achievement of automatic actions that are set-off and executed unconsciously). In human beings, executing a specific task with multiple steps requires to concentrate on the execution of each step individually, until we may become so familiar with the procedure that we can do it almost automatically. In such cases, we accelerate, even automatize the execution and eventually combine steps to reach the final goal more rapidly. We may also have accelerated the decision process that leads to the selection and triggering of an act.

To achieve this, we propose two different kinds of learning in CTS' Behavior Network. We also propose an explicit, high-level learning in the BN that aims at lowering consciousness and attention mechanism involvement, in accordance to a natural phenomenon recently revealed. Transiting know-how to unconscious operations does not just achieve speed gains, but also incurs reduced brain activity, a boon for energy conservation.

Before examining the learning processes, we need to present a few more notions about CTS' architecture. The BN is a high-level procedural memory, a network of partial plans that analyses the context to decide *what to do*, which behavior to set off. This structure is linked to the latent knowledge of *how* to do things in the form of inactive codelets. In fact, External stimuli are interpreted by CTS' perceptual mechanism and written into WM, where it may then be chosen by the attention mechanism to be presented to consciousness. That broadcast information may either assert preconditions for the initiation of a behavior in BN. Or it may cause reaction by another part of the system, which then creates the necessary preconditions for firing a behavior. When a behavior is chosen, it activates the codelets that implement it.

Each behavior node, just as a codelets, has a base-level activation, which can increase or decrease. Until it is selected for execution, a behavior node accumulates energy from the various sources in the BN (feelings, state nodes, other nodes) but they are at the same time submitted to a constant loss of activation. Links between the nodes are concerned with energy too: they learn that way, and they weaken when not used (when the nodes they link are not selected for execution. This mimics human beings: if we do not repeat a task for a while, we will lose some of our ability, forgetting with the passage of time.

We now come to the description of our implicit procedural learning mechanisms.

4.12 First-Level Learning: Learning Through Experience.

In the first level of implicit learning in the BN, two types of reinforcement learning take place: in the links and by the elevation of behavior nodes' base-level activation (Faghihi,U., *et al.*,2007). In the first type, links are reinforced between behaviors that execute in sequence (Figure 4.14.a). This reinforcement makes the BN more goal-driven, more sensitive to the energy coming from the feelings (top-level motivators in the BN) by allowing their *top-down*

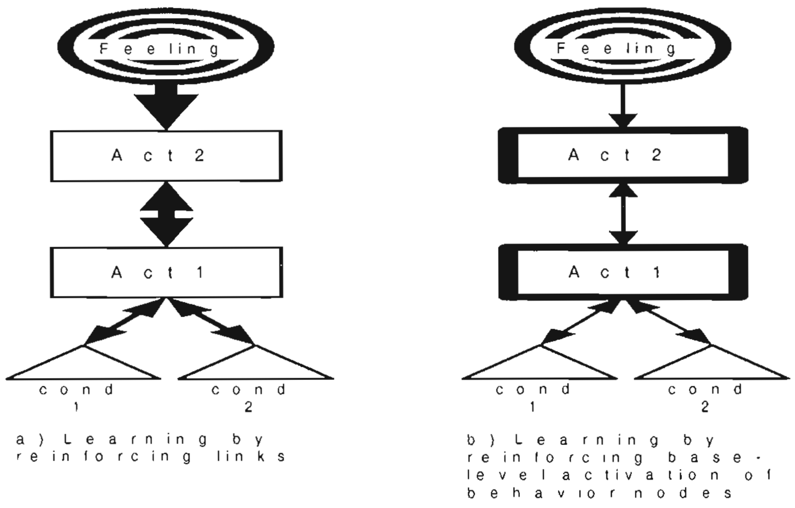

Feeling
Act 2
Act 1
cond 1
cond 2
a) Learning by reinforcing links
Feeling
Act 2
Act 1
cond 1
cond 2
b) Learning by reinforcing base-level activation of behavior nodes

Figure 4.14 Two modes of implicit procedural reinforcement learning for CTS.

stimulation to be amplified when the energy circulates backward in the planning phase (towards the "leafs" of the streams). It results in a faster planning and an increased probability these nodes will be selected for execution next time by being the first ones reaching the activation threshold. The base-level activation in the links is computed with a sigmoid function, the same as for learning of regularities in WM. As a concrete example, if CTS often detects the need for support about planning the path for the payload (refer to Figure 4.15), the sequence concerned with this will fire faster and feel more responsive.

A second type of "*first-level*" learning happens by the elevation of the base-level activation of an act *node* in the BN (Figure 4.14.b) by the intervention of *expectation codelets*. Each action in the BN that is selected for execution releases at least one expectation codelet (our implementation of ideas from Cleeremans, A., *et al*, 1998) that monitors the result of the act. If the result corresponds to the expectation, it is not brought to consciousness (Cleeremans, A., *et al*, 1998) and (Baars, B. J., 1997).The expectation codelet "*silently*" (not going through the conscious broadcasts) sends energy that reinforces the base-level activation of its node. However, if the result did not meet expectations, then inhibition is sent that diminishes the node's base-level energy; the expectation codelet also tries to have the negative result brought to consciousness – these negative results may eventually produce a *suspension-codelet* (some active codelets ,for suspending current task)or induce activation of a repair sequence in the BN. That sort of learning would make the BN more sensitive to the context, more reactive because an act that had success in the past possesses a higher base-level activation and is more likely to be selected for execution on reoccurrence of the same context. When this act receives *contextual* energy (both bottom-up energy coming from an external stimulus and top-down energy coming from *feelings*), its (base-level) pre-activation adds to the energy received; the node will already be more excited than others and has higher probabilities to be selected by the BN Manager.

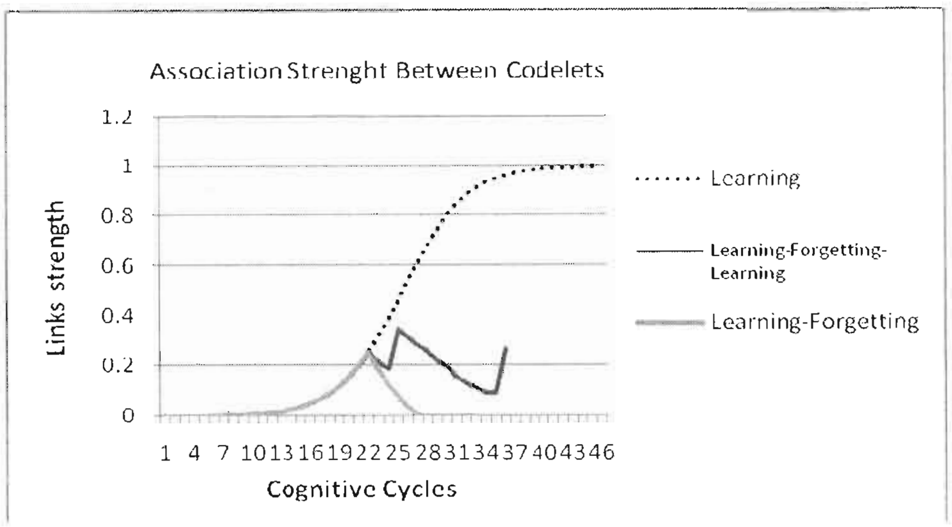


Figure 4.15 Learning and forgetting between behavior nodes in CTS Behavior Network

4.12.1 Second-Level Learning: Partial Automatization.

In human beings, some high-level decisional processes such as decisional selective attention and decisional integration become more efficient with learning (Maddox, W. T., 2002). We have implemented this general idea with a second-level of implicit, learning in BN: partial automatization. Here, not only are links made stronger, but the BN behaves in a different way so that fewer cycles through the access "*consciousness*" are required to complete a task. When the threshold is reached in the link between two act nodes, then the state node between them modifies its behavioral response. It will, from now on, automatically get activated by simply hearing the "*consciousness*"' publication declaring the activation (this happen when nodes preconditions became true) of the behavior it follows, instead of having to wait to hear the expected result of that behavior. This learning is applicable where an operation is broken down in small steps (often for the benefit of the BN designer or those who need to read it later on). It also applies to intermediary states with no meaning that exist only to satisfy the BN design constraints.

When comparing Figure 4.12 and Figure 4.15, one can conclude that in CTS, procedural learning needs less cognitive cycles than perceptual learning. In the same scenario, perceptual learning needs 85 cognitive cycles for sigmoid function saturation, whereas in procedural learning, nodes need only 45 cognitive cycles (black dashed curve) to see their sigmoid function saturate.

4.12.2 Third-Level implicit procedural Learning: Lowering Further Consciousness Involvement (under design).

To execute a new (not *automatized*) task, the brain takes into service a lot of neurons which help by processing the various aspects. After the progressive, iterative learning of the task, it seems that only neurons that realize a direct mapping between the stimulus and the response remain involved (Gilbert, C. D., *et al.*, 2001). When we learn to do something (for example, swimming), in our first attempts, we need to think and respect each step, the way of moving legs, hands, etc. But after a while, we do all of the movements without thinking, without bringing each step into consciousness, bypassing unnecessary explicit processing.

This third-level of implicit learning for CTS (still in the design and development process), offers the possibility for the fusion of extra steps, which will implement the complete idea of efficiency: the system will keeps alive only codelets that are of absolute relevance with the tasks at hand (Gilbert, C. D., *et al.*, 2001). Before learning, the agent needs to execute every step of a procedure consciously. After some reuse of the procedure, there is the possibility for fusion of behaviors (for example fusing Act1 and Act2 in Figure 4.14). Thus after a while, the agent is capable of executing a task "without thinking", thanks to the creation of a new behavior that synthesizes those it replaces. This learning in CTS is possible by creating an appropriate collaboration among Domain Model (DM), Learner Model (LM) and the learning mechanism. However, at this time DM and LM have not implemented completely in CTS. Therefore we cannot experiment this in a concrete manner. As a tutor, CTS can learn a procedure and then decide how to fuse certain acts. This, has not been implemented yet, and could constitute the work of a complete doctoral thesis.

4.12.3 Implicit procedural learning in CTS implementation phase

For implicit procedural learning implementation, as we mentioned above. Learning Mechanisms try to detect all preconditions which will fire a behaviour in the behaviour net.

Or it may generate reaction by another part of the system, which then creates the necessary preconditions for firing a behaviour. The same way as perceptual learning, when Software executed, LM try to bring back all previously activated stated (learned) with their relative codelets which saved in ("C:\\LearningStateParameter.txt") file. Then it makes a temporary copy file for updates and any comparison with actual states and their relative codelets in the Scene. Each node has at least one precondition, some direct links with other codelets and links between codelets have Strengths (the same as perceptual learning mechanism).

```
public class LearningStateFields {

      private State StateName;
      private String StateAssociatedCodelet;
      private double strength;

public LearningStateFields(State sa, String codelet, double
strength){
            this.StateName = sa;
            this.StateAssociatedCodelet = codelet;
            this.strength = strength;
            strengths.add(strength);
            times.add(System.currentTimeMillis());
      }
```

Next codes, task is loading up all previously learned states with their relative codelets into WM.

```
for(Iterator iter = broadcast.iterator();iter.hasNext();){
                  Codelet t = (Codelet)iter.next();
            System.out.println(""+t.getName());
            }
```

Subsequently, some codelets observing the *Scene* tries to register happening events which could be capturing active *states (Pre-conditions)*, *their linked codelets* and the codelets *links* with other codelets. Links between codelets follows perceptual strategy that a link may obtain strength (if it is new in the Scene) or its current strength with others increase/decrease according the time it passes with others in WM.

```
            // LearningStateFields
oldStateValue=LearningStateParameterSaving.getInstance().addNewSta
teLearningStrength(sa);

      //New  getTriggeringCodelet
      s1=sa.getName()+","+sa.getTriggeringCodelet();

      //New State
System.out.println("Add State"+
sa.connections[ConnectionType.ADDITION_CONNECTION .number()]);
```

Now the same as perceptual learning, if codelet is new then we create a list of connections with others active in procedural network. Called codelets into WM will receive more energy for their links, and if they leave WM then, their links with other might lose energy. Loosing energies in the links follows the same scenario as perceptual learning (sigmoid function):

```
double value=oldState.getStrengths().get(chainSize-1);
                  if(value != 0){
// decreaseStateStrength
                        oldState.decreaseStateStrength(value);
                  }
```

And at the end of software execution, all data might be saved:

```
LearningStateParameterSaving.getInstance().saveToFile("C:\\LearningS
tateParameter.txt");
```

4.13 Causal Learning (Underway)

Laura Schulz (Schulz, L. E., and Sommerville, J., 2006) described causal knowledge as crucial and mysterious. Knowing causes, we can change the outcome of situations, which may be crucial. But we have to find relations between events, how they affect outcome; these relations must be inferred. Making inferences can be educated, but the result is never certain; it depends on previous knowledge, available mental tools (or algorithms), and how well these are mastered. It may also depend on the ability to interpret what is perceived. If a group of people see something as a plastic chunk, but others see the same thing as a pen, what's the difference between the first group and the second one? In the first group, people acquire just input/output knowledge without any interpretation. In I/O knowledge, for each input, system allocates an output. In comparison, for the second group, interpretation plays an important role. Interpretation helps to distinguish subjects or events better than simple I/O processing.

Reconstruction of knowledge[8] helps humans to reconstruct the system with its events details in case of doubt. In reality I/O method seems ideal for most application (Lebiere, C., *et al.*, 1998) even in complex systems. ACT-R uses a computational model to acquire knowledge. It uses sub-symbolic form of "*structural knowledge*" to produce *interpreted* data for systems events and "*states*". However by adding causal interpretation, subjects learned "*structural knowledge*" aspects with appropriate associations between system events which called *implicit learning* by author (Wolfgang, S., 2002). In other word, structural knowledge's are useful when no relevant information find about reconstruction of states in the memory (of the system). Though, to gain high performance, using *structural knowledge* mechanism is crucial. *ACT-R system is useful for causal judgment tasks, but it is unable to recognize and interprets interaction between explicit and implicit knowledge for irrelevant stimuli about a new task.* When we need to recognize something it's easier to work with I/O knowledge. For decision tasks, however, structural knowledge is preferable. Wolfgang (in Wolfgang, S., 2002) showed that when an agent forms its knowledge by adding "*Structural knowledge*" information, then this information is useful in interpretation steps and the same when human require constructing a choice according presented causes.

Causal Learning did not implemented in CTS. However, accordingly we plan to implement a similar learning method in the future in our cognitive agent. In Learning of Environmental Regularities, we added strength to the link between two codelets. As we mentioned before, that link may allow the formation of a future coalition if it crosses a strength threshold. Thus, creating new links between codelets and stabilizing those associations that correspond to frequent occurrences accomplish the detection of I/O knowledge. In fact, CTS perceives its environment and in reasoning phase, search through

[8] Structural knowledge (Wikipedia): about how concepts are interrelated, can neither be formally stated nor automatically processed

the saved causes related to the current event happening now. As an example, when CTS', gives lessons to the astronauts, if astronaut is novice and forget a crucial step (camera adjustment), CTS' has to interrupt astronaut by (giving advice, making trouble for him/here, etc). All of this information will save as an Acyclic directed graphs (A Bayesian network[9] which is "*a calculus for rational belief assignment*") for next utilization; our agent might detect, understand and remember such a similar situation with their causes. Griffiths (Griffiths, T.L., *et al.*, 2004), showed some examples of causal learning using the Baysien approach to make the next decision faster without using the same resources and processes or use these sort of causes to make unseen situation (*Reasoning*). Each time system approves a cause (Figure 4.16), links value between codelets related to the current event will receive more energy and learning mechanism will assign to the current event one (or some) proved causal Codelets automatically. Proved cause is saved into a *Causal Memory* (which is *in relation with Conceptual memory*) when the access consciousness broadcasts the information. Thus, during a task execution, when situation is examined, a mechanism reviews the *Causal Memory* to find similar situations that happened before. If found, such events already happened, then associated Codelets will receive more energy. New energy will update into the *Causal Memory*. If system cannot find the same or similar events (that happened before) new event with its associated causes (Codelets) will be created in the Causal Memory.

[9] A causal Bayesian network(Wikipedia) is a Bayesian network where the directed arcs of the graph are interpreted as representing causal relations in some real domain. The directed arcs do not have to be interpreted as representing causal relations; however in practice knowledge about causal relations is very often used as a guide in drawing Bayesian network graphs, thus resulting in causal Bayesian networks.

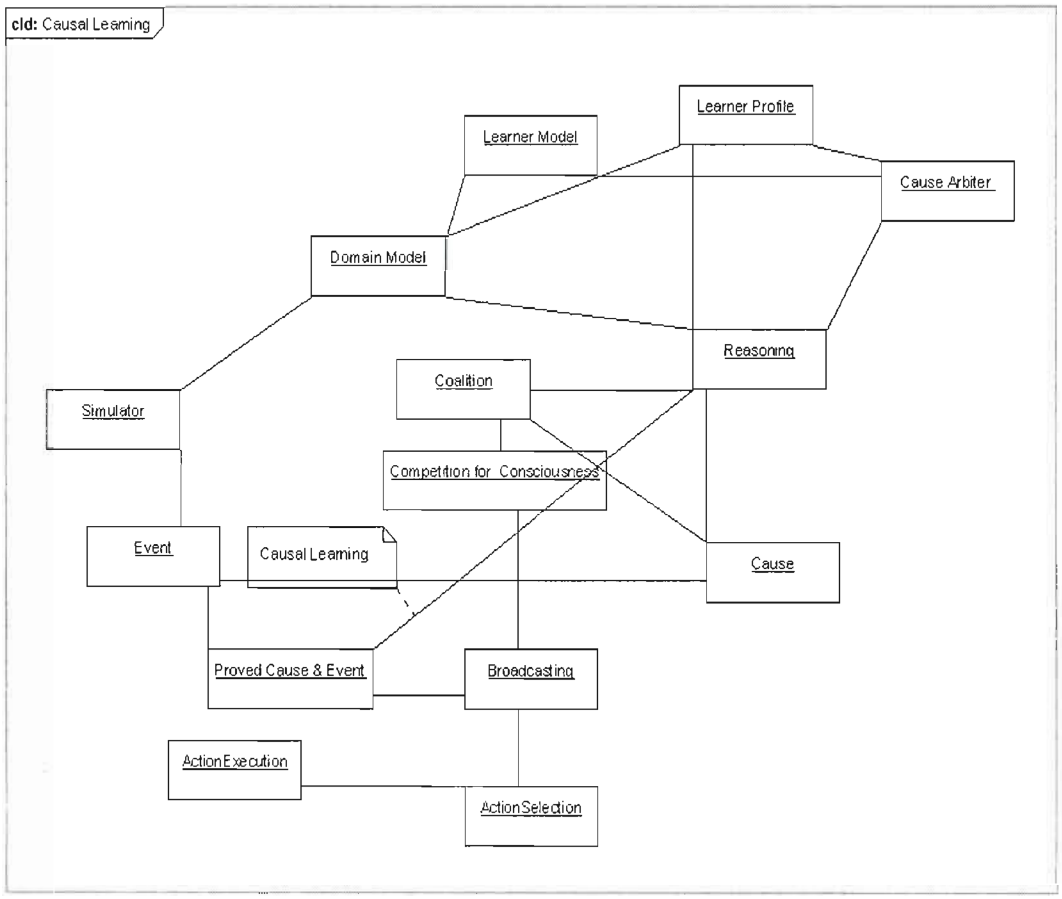


Figure 4.16, CTS Causal Learning Mechanism

In this way, the causal interpretation for each link in causal memory appears into a coalition. In addition, coalitions with proved causes gain more precise *knowledge* and additional *energy* than others for consciousness broadcasting phase in CTS, which helps different modules answer correctly to the broadcasted information. We propose a combination of observation and intervention to learn causal interpretation in which, it conduct us to causal learning. System can estimate the probability of each links between two codelets, to insist in the next coalition by Baysian[10] net formalism.

10 Wikipedia:A causal Bayesian network is a Bayesian network where the directed arcs of the graph are interpreted as representing causal relations in some real domain. The directed arcs do not have to be interpreted as representing causal

For example:

`Codelet-Info-Collision-*` in WM caused presence of

```
Codelet-Info-Station-Element-S1P1TrussRight01

Codelet-Info-Distance-*

Codelet-Info-Joint-WE
```

It means Collision, appear when a joint (Codelet-Info-Joint-WE) approach's to one of the SSI elements in the space. The cause could be astronaut distraction, fatigue or etc, which current causes might be coded as a part of coalition to be used in next or similar event.

relations; however in practice knowledge about causal relations is very often used as a guide in drawing Bayesian network graphs, thus resulting in causal Bayesian networks.

CHAPTER V

COMPARISON WITH OTHER WORKS

In this chapter, we will compare our architecture with three other works focusing on their learning dimensions. It consists of a comparison with Franklin architecture, which we raise and justify the modifications (for learning mechanism). Then, we compare three popular architectures as BDI Agents (a general architecture of agent), CLARION (a cognitive agent), ACT-R (aspire to be general computational models of cognition)

5.1 Comparison with LIDA architecture

I organized a brief comparison between LIDA and CTS learning aspects in table 5.1.

LIDA (Franklin, 2006)	**CTS(2006)**
Perceptual Learning(Explicit)	---------------
Episodic Learning	--------------
Attention Learning	Learning of Environmental Regularities (Implicit/Explicit)
Procedural Learning(Explicit)	Procedural Learning(Implicit/Explicit)
	Meta-Cognition Learning(fusion of actions) [Underway]
	Emotional Learning [Underway]

Table 5.1, Comparison between LIDA and CTS.

1- The first difference between CTS and LIDA is that we actually have no perceptual or episodic learning in our agent. As all entities that produce events (danger, motion, menu, etc) in virtual world are predefined in CTS, which was not a priority in building our prototype. CTS' perception functionality is different from LIDA's. LIDA use a *Slipnet* mechanism to understand and give the premier meaning to the received information. In comparison, CTS uses a simpler semantic network.

Presently our group is working on various memory aspects (currently exclusively based on the "*Sparse Distributed Memory*" algorithm, as in LIDA) to give episodic learning capability to the agent.

2- We propose unconscious learning for the learning of environmental regularities (based on Jackson's theory and on real experimentations with humans by psychologists); LIDA uses just consciousness for learning (based on Baars Global Workspace theory). In fact, we propose parallel tasks accomplishment by CTS and at the same time different type of learning. We implemented a way that when agent executes a special task, at the same time, it can receive information from its virtual world, and learn in parallel (Faghihi,U., *et al* .,2007).

3- Implicit procedural learning,

3-a) our method to accomplish automatization of routine processes is different. Franklin proposed a way to automatize the execution of behaviour streams (Franklin, S., 2005). Although it is founded on Jackson's ideas, it relies on the intervention of attention codelets that progressively fade out of the repeating process. We implement the automatization of streams around the same idea of repetition of usage, but in the higher-level structure, the BN. Links are reinforced between behaviour nodes instead then between codelets. Our approach remains somewhat further to the biological model but is able to reproduce the phenomenon of faster reactions with practice. In our manner of doing, the BN becomes then more goal-driven, more sensitive to the energy coming from the feelings. Then agent executes and achieves its goal faster. By this notion, CTS is more adaptable to the changes in its environment.

3-b) Implicit bias towards successful behaviours

LIDA as well as CTS reinforce their successful behaviours. In LIDA, it happens through it mechanism that learns new schemes or reinforces an existing one's base-level activation (D'Mello, S. K., *et al.*, 2006). They describe the role of expectation codelets in this mechanism. CTS also use expectation codelets for the same goal, but in a way that differs on two accounts. First, whereas LIDA sees its expectation codelets try to bring back to consciousness the result of the action, experimentations of Curran and Keele (Curran, T., Keele, and S.W., 1993) proved that in procedural tasks, human beings execute the task and learn at the same time without necessarily bringing the result of each action into consciousness. Action results are brought back into consciousness when they are wrong, except if voluntary attention is devoted to awaiting the result. Of course, the second difference is that reinforcement of successful behaviours happens thanks to conscious broadcasts. In CTS, reinforcement is accomplished "*silently*", at the unconscious level, not making use of conscious broadcasts.

3-c) The brain in human beings, to accomplish a procedural task, allocates some related and unrelated neurons. After the few first executions, the brain activity will decrease. It means that just related neurons for accomplishing the task will remain active, thus CTS might accomplishes the task and learn them without using the same (quantity of) resources that were allocated in the first executions.

5.2 Comparison with BDI architecture

The information about the environment in the BDI architecture is described as beliefs. Desires indicate the set of current goals that agent wants to attain; desires are reciprocally exclusive. One of the intentions is selected to become the active desire. There is a library of plans defined in BDI model. Any change in the environment will be reflected as events. Event-queue saves the order of events that happened during the implementation of plans in the agent.

In CTS, information (represented by information codelets) on the Scene and active States are our representation of the world. Thus, they correspond to the agent's beliefs. The Feelings and *Desires* (with our meaning of the word) nodes of the procedural network are similar to the initial *intentions* (with our meaning of the word) of CTS, and they guide the choice of other "sub-"intentions. Feeling and Desire nodes, because they are not used to satisfy another intention, are top-goals in themselves, but until an action is selected in the BN and fires (thus becoming an intention (BDI meaning)), they remain in the role of "desires" (BDI meaning). The behaviour nodes in the Behavior Nėtwork may consecutively take the role of BDI desires and BDI intentions.

Being a cognitive architecture that implements a richer version of BDI, CTS architecture is better adapted to an intelligent tutorial system than the plain BDI model. Perception may make only fast recognition of the situation, it detecting whether there exists a collision or any other pressing danger. That may trigger an alarm reaction that bypasses all the deliberative processes. Later on, the danger will be broadcasted in theatre with information codelets bringing more information on the scene.

As soon as this information is broadcasted, the agent will continue to interpret the situation by revision of its memories ("If this problem happened before?"), different modules which detect broadcasted information will answer. Next time, if agent perceives the same or a similar situation, the same codelets will be invoked, and transfer to consciousness. Consequently, the answer will happen faster than in previous invocations. CTS', learns step by step with reinforcement that come from environment. Furthermore, CTS can internally revise all of its actions and can modify them if needed (when we implement the necessary behaviour streams in the BN). CTS is capable to distinguish priorities of goals set by its various parts (Learner and Domain models, among others) and select the most important, urgent, and relevant event.

5.3 Comparison with ACT-R

ACT-R (1990) architecture implements human cognition model. ACT-R performances are frequently compared with that human beings reaction to check if the architecture produces similar results (Anderson, J. R., and Lebiere, C., 2003).

ACT-R is the best-validated simulation of human beings cognition theory (ACT–R; e.g., Anderson, J. R., and Lebiere, C., 1998). ACT-R uses a modular architecture which consists of a central part with a set of buffers that permit *indirect* communication between different modules in the system.

ACT-R consists of different modules (Figure 5.1) such as a perceptual (*"visual"*) module for recognizing the objects, a goal module who's task is current goal and intention pursuing, declarative memory module for recovering information from the memory and procedural module for controlling agent's movements (or actions, in general). In this architecture, modules cannot communicate directly: any communication must pass through *"central production system"*. Each buffer contains one *declarative* piece of facts, called a "*chunk*". Such a "*chunk*" consists of a name and labelled links towards others "*chunks*", this forming a "*semantic network*". The inference module modifies the content of buffers following a set of rules called "*productions*".

Each *production rule* is composed of conditions (which indicate to which configuration, or content, of the buffers it is applicable) and actions (indicates how it modifies the buffers). ACT-R uses the production rules to solve procedural problems (for example a mathematical subtraction). These rules are specific to the application, but ACT-R provides meta-rules to choose and execute a particular rule, because in each cycle there is only one rule that can be carried out by the system. Cognitive cycles in ACT-R start by finding a pattern for external or internal image of the world which corresponds to the buffers; then a production rule is triggered, and then buffers will be updated for the next cognitive cycle. This complete cycle length is estimated at about 50ms; in LIDA the same cycle is granted about 200ms.

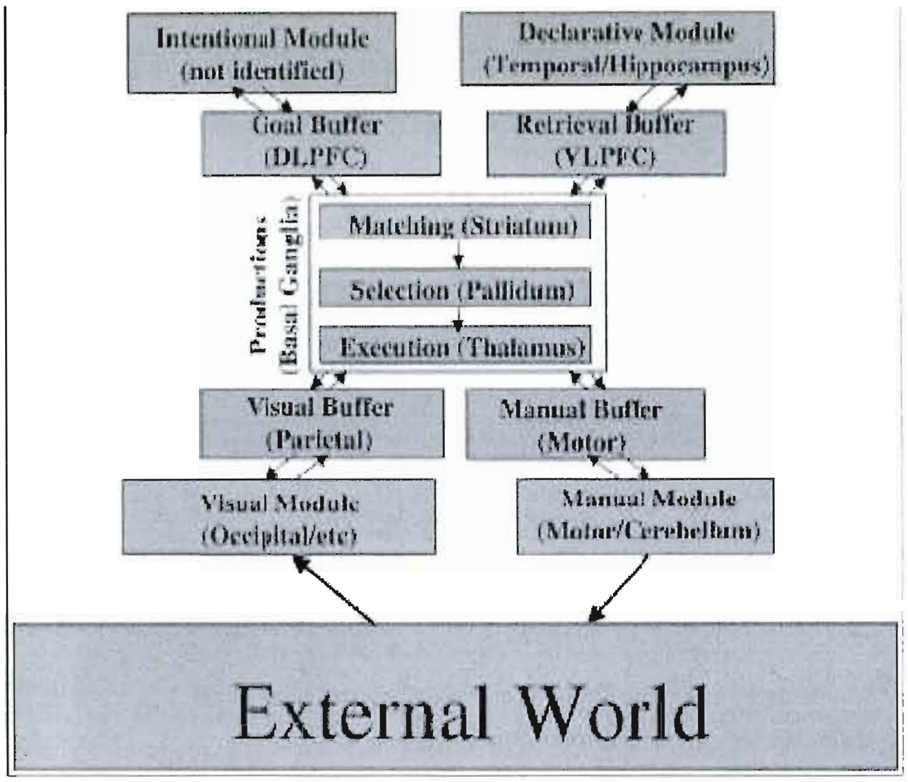

Intentional Module (not identified)
Declarative Module (Temporal/Hippocampus)
Goal Buffer (DLPFC)
Retrieval Buffer (VLPFC)
Productions (Basal Ganglia)
Matching (Striatum)
Selection (Pallidum)
Execution (Thalamus)
Visual Buffer (Parietal)
Manual Buffer (Motor)
Visual Module (Occipital/etc)
Manual Module (Motor/Cerebellum)
External World

Figure 5.1, ACT-R 5.0 Architecture

Learning in ACT-R is possible for both of *chunks* and *production* rules at the symbolic and Sub-symbolic level ("*knowledge structure*"). Implicit learning in ACT-R is possible by a particular *declarative chunk* that can be fetched and examined and explicit learning could be the results of learning goals .The chunk resulted from a goal (the solution), will gain its *base-level* energy and store into declarative memory. Next time, when system called the same task, the result according to the previously stored activation of chunk will be recovered from declarative memory instead of recalculating it again.

$$A_i = B_i + \sum_j W_j S_{ji}$$

In this formula, B_i is the *base-level* activation of the chunk i, W_j shows the sum of noticed weighting for each components j which currently detected by the attentional part of the system, and S_{ji} depicts the strength of association from each component j.

Sometimes a goal could be achieved through a complex process, and then a "*dependency goal*" will create by the system to understand and save how the problem solved. Next time when the same or similar goal happens, the "***Production rule***" is accessible to solve the problem in a single step.

Sub-symbolic Learning, happen, when system uses production rules to attain a goal (useful experiences). The target here is making existing symbolic knowledge more available. The chunk which is often used, become more active. Production parameters will be updated by experience (it could increase/decrease the probability of happening such a production in the future). Each time a chunk called its base-level will increases and the strength of association between the current sources and the chunk is also increased.

As a brief comparison between ACT-R and LIDA, ACT-R does not focus on consciousness as much as LIDA. ACT-R is not equipped with an *episodic memory*. Learning in LIDA happens after information is broadcasted by Consciousness. It means agent learn when attention mechanism bring appropriate information to the consciousness. In fact, LIDA learn explicitly, but, ACT-R learn both implicit and explicitly.

Meta-cognition was not addressed in ACT-R.

ACT-R symbolic and sub-symbolic incorporation is similar to LIDA. However, ACT-R symbolic level is focused on *production rules* and *chunks*, whereas the sub-symbolic level is made of various parallel processes modeled by a collection of equations. One of ACT-R limits is its difficulty concerning development and implementation by new users. Development phases in ACT-R, need raising a complicated model of the cognitive task. ACT-R did not address the social impact of the software and is not a "*generative*" theory. ACT-R is unable to explain the *ascending learning* of explicit knowledge and the interaction between explicit and implicit knowledge. In ACT-R, the rules for all situations must be specified in advance. ACT-R did not answer that in implicit learning what type of knowledge could be learned?

Remaining questions about ACT-R architecture are:

1. In ACT-R ,"*In the symbolic knowledge representation* , maybe it is more acceptable to use symbol grounding or symbol tethering to create a better representation of the world (*physic and social*)" .

2. Another question about ACT-R architecture is: "How could we model language grammars that construct from rules, sub-rules, exceptions with a clear distinction and justification between different categories".

3. "The most important part of expertise in human being is intuitive, immediate and unconscious, then *is it truthful, valid, to produce human cognitive processes as a production system, which are explicit and precise*?". Production rules are simply the expressive units and not more.

4. Human cognition operates on a complex social environment and is situated in the real world. Creation of cognition needs social interactions, cultural background, common language, and etc. ACT-R did not address these important issues in its architecture.

5.4 Comparison with CLARION

CLARION (*Connectionist Learning with Adaptive Rule Induction ON-line*) (Ron Sun, *et al.* 1998, 2001, 2005), is aimed at obtaining various cognitive processes within a single cognitive architecture. It is a cognitive modules-based agent. It investigate interactions between different parts with respect to procedural and declarative knowledge) of an agent by integrating *connectionist*, *reinforcement*, and *symbolic* learning methods to obtain several learning abilities, such as "*bottom-up learning*[11]", "*trial-and-error learning*", "*top-down*

[11] Bottom-up learning: Learning that goes from implicit to explicit knowledge (Ron Sun et al, 2004).

learning", using "*cognition-motivation*" when the agent interacts with its environment ("*meta-cognitive interactions*"). Ron Sun proposed a middle level between "*phenomenology*[12]" and "*physiology/neurobiology*" for encapsulating basic characteristics of consciousness.

The most important challenge in the CLARION architecture is interactions between implicit and explicit knowledge that an agent acquire from its environment. To represent both implicit and explicit knowledge, the architecture uses two separate parts. In fact, different abilities/expertise (procedural knowledge) can be obtained by a back-propagation[13] network. However, the implicit characteristics of procedural abilities/expertise, features of such abilities/expertise are not accessible to consciousness (Anderson, J. R., 1983; Reber, A. S., 1989). In this architecture, explicit (declarative) knowledge is achieved through a symbolic representation by computational features of the system. Such information is more accessible to consciousness. Ron Sun proposed a distributed system with sub-systems that all separate implicit knowledge from explicit knowledge. Each sub-system has two levels; top level encodes explicit knowledge and the bottom level encodes implicit knowledge.

In figure 5.2, ACS (the "action-centered sub-system") controls internal and external actions. NACS ("*non-action-centered subsystem*") role is to support explicit and implicit knowledge about the world, in which performs as a motivator and gives correspondent feedback to different part of the system (perception, action, cognition, etc). MS is the motivational subsystem for feedback proposes. MCS ("*meta-cognitive subsystem*") observes ACS and all the other sub-systems of the agent, their activities and operations in order to change them when needed (for example, when new feedback is received). CLARION is equipped with a procedural memory, a declarative memory and an episodic memory.

In the other word, in this architecture top level encodes explicit declarative knowledge; bottom level encodes implicit procedural knowledge.

[12] Phenomenology(answers.com): *A philosophy or method of inquiry based on the premise that reality consists of objects and events as they are perceived or understood in human consciousness and not of anything independent of human consciousness.*

[13] Back-propagation (Wikipedia) :*A common method of training a neural net in which the initial system output is compared to the desired output, and the system is adjusted until the difference between the two is minimized.*

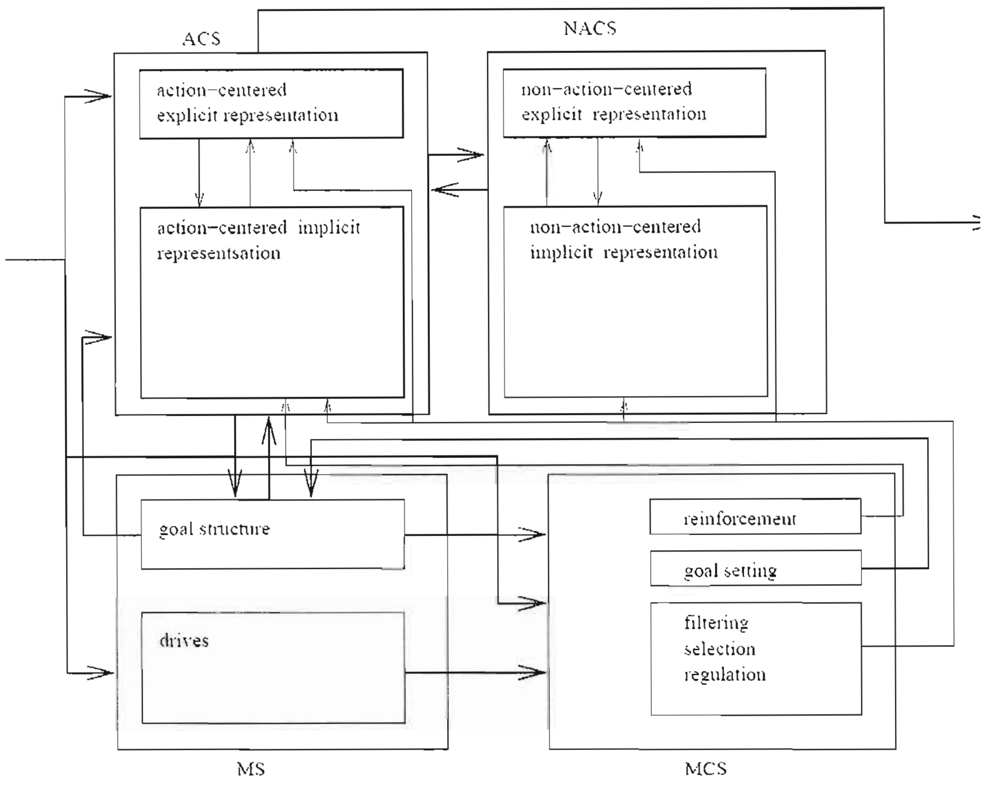

ACS
NACS
action-centered explicit representation
action-centered implicit represeutsation
non-action-centered explicit representation
non-action-centered implicit representation
goal structure
drives
reinforcement
goal setting
filtering
selection
regulation
MS
MCS

Figure 5.2, CLARION architecture (Anderson 2004)

In this architecture, cognition performs as a bridge between the needs and motivations of agent with its environments. The model and its learning processes enable an autonomous learning mechanism. Learning is accomplished by the integration of reinforcement learning and rule induction, so that the resulting process is included automatically in the structure. Implicit learning ("*bottom-up*") happens at the bottom level with supervised learning ("*back-propagation network*") by adjusting input/output parameters.

Explicit learning happens by turning acquired knowledge from implicit knowledge into symbolic representations. In fact, explicit knowledge is an extraction and refinement of information that was captured from interaction with environment (implicit knowledge). Conversely, explicit knowledge will be integrated into the bottom level after it becomes stable.

The agent explores its environment and tries to acquire information or modify it (for example, hypothesis testing without the help of bottom level). The action selection mechanism in CLARION is formed by different top/bottom levels. There exist input and output for both levels. A state entered from the environment into the system will be analysed at first, and then an appropriate action will be allocated, according to the considered goal. The feedback will be learned and saved for future uses. In fact, the feedback could be translated into "*rules*" and "*chunk*". Each action took by the bottom level will produce a node with some related rules in the top level after extraction of explicit rule and then it will refined by future interactions with external world. Some existing node in the bottom level may be relevant to the condition of rules that linked to a sole node at the top level which could indicate condition recognized as a chunk node. Another important characteristic which may be used for social behaviours is motivational aspects. Without it, the agent is purposeless. With this mechanism, the agent is capable of creating automatically proper feedback from environment without extra coding or receiving feedback from its environment.

A comparison between CLARION and Baars' model shows that CLARION takes into account the integration of inputs from sensory or devices, global reliability, and integrity of *consciousness*. As explained in Baars' (1988) model, a large number of Codelets accomplish unconscious processing, and the global workspace synchronizes and manages their activities through a broadcasting overall the system. This model carries some similarity to CLARION. Unconscious *Codelets* could be considered as the bottom-level of CLARION; and global workspace could be considered as its *top-level* which "*synthesizes*" *bottom-level* modules. CLARION does not work as much as Baars to internal uniformeness and its architecture has partial functionality in the emersion of consciousness (Marcel, A., 1983).

Global broadcasting in Baars' model could be considered as the incorporation between *top* and *bottom* level representation, scattered within multiple modules in CLARION that will conduct to the unity of consciousness. However, various learning mechanisms (*episodic*, *attention*, *emotional*) have not addressed directly by CLARION. At the theoretical level, CLARION uses a supervised *connectionist* network for the procedural module (*implicit*), that is, rests on feedbacks to accomplish its bottom-level learning, whereas experiments with human beings relating to *implicit learning* usually do not provide feedback to the subjects

(Cleeremans, A., 1997; Cleeremans, A., *et al.*, 1998; Curran, T., Keele, S.W., 1993). Maybe, it is better to use *non-supervised connectionist* sub-*symbolic networks* for procedural modules (Hélie, S., 2007). CLARION uses distributed representations to represent implicit knowledge, whereas these latter should be characterized by sub-symbolic mechanism.

5.5 Comparison between different architectures

At this point, we compare our architecture with three popular architectures explained briefly in this chapter.

LIDA (Franklin, 2006)	**ACT-R**	**CLARION**	**CTS(2006)**
Perceptual Learning(Explicit)	Explicit Learning	Explicit Learning	Learning of Environmental Regularities (Implicit/Explicit)
Episodic Learning	Implicit Learning	Implicit Learning	Procedural Learning (Implicit)
Attention Learning	---------------	Meta-Cognition Learning	---------------
Procedural Learning	---------------	---------------	---------------
---------------	---------------	---------------	Emotional Learning [Underway]

Table 5.2, Comparison between LIDA, ACT-R, CLARION and CTS.

CHAPITRE VI

CONCLUSION

The CTS architecture is the product of an on-going research of the GDAC group (Research laboratory for knowledge management, dissemination and acquisition) at UQAM (Université du Québec à Montréal). CTS is a complex software agent.

The cognitively-oriented software agents explained in this survey are helping us comprehend the notions underlying learning of environmental regularities and procedural learning. The learning of environmental regularities in CTS is inspired from Jackson's extension to Selfridge's Pandemonium theory. Our implementations are inspired by both *selectionist* and *instructionalist* approaches. They respect what is generally accepted knowledge about psychological and biological mechanisms (parts of Baars' neuro-psychological theory, Cleeremans' research). The equations that drive learning in CTS are based on well established studies (Hebbian learning and forgetting curve). Furthermore, they happen concurrently in our agent. We believe that if we want to attain humanly levels of performance in general intelligence, it is important that we stay close to neuropsychological studies, even is by iteratively reaching that goal in successive efforts.

We offered two kinds of incremental learning mechanisms that place the basis for a cognitive architecture capable of human-like learning at theoretical, design and computational levels. In fact, learning in CTS could be summarized as a combination of low-level (continuous) learning with high-level (“symbolic") learning of entities relationships.

These learning mechanisms will allow CTS to better adapt to its environment and perform its tasks more swiftly. By integrating emotional learning in CTS, we expect that new knowledge easily join together with the old one. Ergo, parallel tasks accomplishment and learning might happen faster, should become easier for CTS, and make the agent more reliable.

We think that CTS' model could be helpful as an instrument to sustain cognitive scientists and neuroscientists in their research by producing testable hypotheses which can be analyzed to either confirm or deny some human cognition theories. We have shown a combination of continuous (implicit) and symbolic (explicit) models as a powerful paradigm which will open up many further avenues of research over the coming years. CTS' architecture is partially implemented but already sustains a working model of cognition, as fundamentally based on Franklin's LIDA architecture. We intend to run experimentations to evaluate improvements expected to CTS behavior during astronauts' training sessions. We expect faster reaction times to help them not create dangerous situations (collision, etc.)

Testing and proving all of the proposed models, all their parameters, against cognition theories and experimental results are impossible within the timeframe of my master research. Since our application is a tutoring agent, performance should be evaluated on the basis of improvement of the tutorial actions coming from the learning mechanisms put in place. However, the tutoring capabilities are still very limited in our prototype and cannot show so well the potential of these learning mechanisms. In the pursuit of CTS project, our team plan to develop other important learning mechanisms: emotional learning, meta-cognitive learning, episodic learning, attention learning. Some of them are already underway for LIDA, but since our implementation differs, we cannot reuse their efforts directly. We will have to keep a parallel research and development effort, sharing only at the conceptual level.

Bibliography

Aamodt, A., and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues,Methodological Variations, and System Approaches. *AI Communications, Vol. 7 Nr. 1, March* , 39-59.

Albus, J. S. (1991). Outline for a Theory of Intelligence. *IEEE Transactions on Systems, Man and Cybernetics, Vol. 21, No. 3, May/June.*

Anderson, J.R. (1983). The Architecture of Cognition. Cambridge, MA: Harvard University Press.

Anderson, J. R., and Lebiere, C. (1998). The atomic components of thought. *Mahwah, NJ: Erlbaum.*

Anderson, J. R., and Lebiere, C. (2003). The Newell test for a theory of cognition. *Behavioral and Brain Sciences, 26, 587-640.*

Anwar, A., and Franklin, S. (2003). Sparse Distributed Memory for "Conscious" Software Agents. *Cognitive System s Research 4:339-354.*

Baars, B. J. (1988). A Cognitive Theory of Consciousness. *Cambridge: Cambridge University Press.*

Baars, B. J. (1983). Conscious contents provide the nervous system with coherent, global information. *In R.J. Davidson, G.E. Schwartz & D. Shapiro (Eds.).Consciousness & Self- regulation.* New York, NY: Plenum Press.

Baars, B. J., and Franklin, S. (2003). How conscious experience and working memory interact. *Trends in Cognitive Science 7:166-172* .

Bogner, M., Ramamurthy, U., and Franklin, S. (2000). Consciousness" and Conceptual Learning in a Socially Situated Agent. *In Human Cognition and Social Agent Technology, ed. K. Dautenhahn. Amsterdam: John Benjamins.*

Bratman, M. E., David, J. I., and Martha, E. P. (1998). Plans and resource-bounded practical reasoning . *Computational Intelligence, vol. 4, no 4.*

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation, vol. 2, no 1, pp. 14-23.*

Cheesman, J., and Merikle, P. M. (1984). Priming with and without awareness. Percept. Psychophys., 36, pp. 387-395.

Cleeremans, A. (1997). Principles for implicit learning. In: Berry, D. (Ed.), How implicit is implicit learning?. *Oxford University Press, Oxford, UK* , pp. 195-234.

Cleeremans, A., and Jiménez, L. (1996). Implicit cognition with the symbolic metaphor of mind. *Theoretical and methodological issues (unpublished manuscrit).*

Cleeremans, A., and Jiménez, L. (2002). Implicit learning and consciousness: A graded dynamic perspective" Implicit learning and consciousness: An empirical, computational and philosophical consensus in the making?". *French, R. M. and Cleeremans, A. (Eds.), Psychology Press, Hove, UK.*

Cleeremans, A., and Jiménez, L. (1998). Implicit sequence learning : The truth is in the details. In M. A. Handbook of Implicit Learning. Sage Publications, Thousand Oaks, pp. 323-364.

Curran, T., and Keele, S. W. (1993). Attentional and nonattentional forms of sequence learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 19, 189-202.*

Davis, D. N. (2002). Architectures of cognitive and a-life agents. *Neural, Emergent and Agent Technology Research Group.* Department of Computer Science ,University of Hull: Kingston-upon-Hull.

Dienes, Z., and Berry, D. C. (1997). Implicit learning: Below the subjective threshold. *Psychonomic Bulletin & Review, 4, pp. 3-23.*

D'Inverno, M., and Kinny, D. (1997). A Formal Specification of dMars. *In Agent Theories, Architectures, and Languages(Ed.)*, (pp. pp.155-176).

D'Mello, S. K., Franklin, S., Ramamurthy, U., Baars,B. J. (2006). A Cognitive Science Based Machine Learning Architecture. *AAAI 2006 Spring Symposium Series. Ameri-can Association for Artificial Intelligence, Stanford University, March 2006.*

Drescher, G. L. (1988). Learning from Experience Without Prior Knowledge in a Complicated World. *Proceedings of the AAAI Symposium on Parallel Models. AAAI Press.*

Drescher, G. L. (1991). Made-Up Minds: A Constructivist Approach to Artificial Intelligence. *Cambridge, MA: MIT Press.*

Dubois, D. (2007). *Constructing an agent equipped with an artificial consciousness:Application to an intelligent tutoring system.* PHD Thesis.

Ebbinghaus, H. (1885). A Contribution to Experimental Psychology. *Über das Gedchtnis. Untersuchungen zur experimentellen Psychologie.* New York: Teacher College, Columbia University.

Faghihi, U., Dubois, D., and Nkambou, R. (2007) Learning Mechanisms for a Tutoring Cognitive Agent. Proccedings of the 7th IEEE International Conference on Advanced Learning Technologies (ICALT 2007), July 18-20, Niigata, Japan

Ferguson, I. A. (1995). On the role of DBI modeling for integrated control and coordinated behavior in autonomous agents. *Applied Artificial Intelligence, 9(4).*

Franklin, S. (1995). Artificial Minds. *Cambridge MA: MIT Press .*

Franklin, S. (2005). Cognitive Robots: Perceptual associative memory and learning. *In Proceedings of the 14th Annual International Workshop on Robot and Human Interactive Communication (RO-MAN 2005).*

Franklin, S. (2000). Deliberation and Voluntary Action in 'Conscious' Software Agents. *Neural Network World 10:505-521.*

Franklin, S. (2003). *IDA: A Conscious Artifact.* Journal of Consciousness Studies 10: 47-66.

Franklin, S., and Ramamurthy, U. (2006). Motivations, Values and Emotions: Three Sides of the same Coin. *Proceedings of the Sixth International Workshop on Epigenetic Robotics, Paris, France, September 2006,.*

Franklin, S., Baars, B. J., Ramamurthy, U., and Ventura, M. (2005). The Role of Consciousness in Memory. *Brains, Minds and Media*, (pp. 1:1-38, pdf).

Gilbert, C. D., Sigman, M., and Crist, R. E. (2001). The neural basis of perceptual learning. *Neuron, 31 (5), pp. 681-697.*

Griffiths, T. L., Baraff, E. R., and Tenenbaum, J. B. (2004). Using physical theories to infer hidden causal structure. *Proceedings of the Twenty-Sixth Annual Conference of the Cognitive Science Society.*

Hayes-Roth, B. (1995). An architecture for adaptive intelligent systems. *Artificial Intelligence, 72 329-365.*

Hebb, D. O. (1949). The organization of behavior. Wiley, New York.

Hélie, S. (2007). *Modélisation de l'apprentissage ascendant des connaissances explicites dans une architecture cognitive hybride.* PHD Thesis, Montreal.

Hofstadter, D. R., and Mitchell, M. (1994). The Copycat Project: A model of mental fluidity and analogy-making. *In Advances in connectionist and neural com putation theory, Vol. 2: logical connections, ed. K. J. Holyoak, and J. A. Barnden. Norwood N.J.: Ablex.*

Inman, J., and Hewitt, C. (1991). DAI Betwixt and Between: From "Intelligent Agents" to Open Systems Science. *IEEE Transactions on Systems, Man, and Cybernetics. Nov./Dec. 1991.*

Jackson, J. V. (1987). Idea for a Mind. *Siggart Newsletter, 181:23-26.*

Johnson, M., and Scanlon, R. (1987). Experience with a Feeling-Thinking Machine. *Proceedings of the IEEE First International Conference on Neural Networks, San Diego.71-77.*

Kanerva, P. (1988). Sparse Distributed Mem ory. *Cambridge MA: The MIT Press.*

Kelemen, A., Liang, Y and Franklin, S. (2002). A Comparative Study of Different Machine Learning Approaches for Decision Making. *In Recent Advances in Sim ulation, Com putational Methods and Soft Com puting, ed. E.Mastorakis. Piraeus, Greece: WSEAS Press.*

Lebiere C., Wallach, D. and Taatgen, N. (1998). Implicit and explicit learning in ACT-R. *In F.E. Ritter & R.M. Young (Eds.), Proceedings of the Second European Conference on Cognitive Modelling (ECCM-98), pp. 183-189. Nottingham: Nottingham University Press.*

Maes, P. (1989). How to do the right thing. *Connection Science 1:291-323.*

Marcel, A. (1983). Conscious and unconscious perception: an approach to the relations between phenomenal experience and perceptual processes. *Cognitive Psychology*, (pp. 15, 238–300.).

McCauley, L., and Franklin, S. (2002). A Large-Scale Multi-Agent System for Navy Personnel Distribution. *Connection Science 14:371-385.*

Miyake, A., and Shah, P. (1999). Models of Working Memory. Cambridge. *Cambridge University Press.*

Negatu, A., and Franklin, S. (2002). An action selection mechanism for 'conscious' software agents. *Cognitive Science Quarterly 2:363-386.*

Newell, A. (1990). Unified Theory of Cognition. *Cambridge, MA: Harvard University Press.*

Nkambou, R., Belghith, K,. and Kabanza, F. (2006). *An Approach to Intelligent Training on a Robotic Simulator using an Innovative Path-Planner.* LNCS No.4053, pp. 645-

654, Springer-Verlag, Berlin.: Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS'2006).

Reber, A.S. (1989). Implicit learning and tacit knowledge. Journal of Experimental Psychology: General, 118, 219-235.

Schulz, L. E., and Sommerville, J. (2006). Causal Determinism and Preschoolers' Causal Inferences, Child Development. 77(2), 427-442.

Selfridge, O. G. (1959). Pandemonium: A Paradigm for Learning. In: Proceedings of the Symposium on Mechanisation of Thought Process. *National Physics Laboratory.*

Sloman, A. (2001). Evolvable Biologically Plausible Visual. *The University of Birmingham.* School of Computer Science.

Sloman, A. (1995). Exploring Design Space and Niche Space. *Proceedings 5th Scandinavian Conf on AI, Trondheim May 1995, Amsterdam: IOS Press.*

Sloman, A., and Chrisley, R. (2003). Virtual machines and consciousconsciousness. *In Journal of Consciousness Studies* , vol.10, nos (4-5), pp. 133-172.

Sun, R. (2004). Philosophical Psychology. *Desiderata for cognitive architectures* , 17, 341-.

Turing, A. M. (1936). On computable numbers with an application to the Entscheidugsproblem. *Proceedings of the Mathematical Society: Série* 2, *42*, 230-265.

Vidal, H. J., and José, M. (2006). From Rational to Emotional Agents. *Proceedings of the AAAI Workshop on Cognitive Modeling and Agent-based Social Simulation.*

Wolfgang, S. (2002). Stochastic Independence between Recognition and Completion of Spatial Patterns as a Function of Causal Interpretation. *In Proceedings of the 24th Annual Conference of the Cognitive Science Society.*

Wooldridge, M. (1999.). « Intelligent Agents ». Dans Weiss, G., editor: Multiagent Systems,.

Zhang, Z., Franklin, S., Wan, Y. O. B., and Graesser, A. (1998). Natural Language Sensing for Autonomous Agents. *In Proceedings of IEEE International Joint Sym posia on Intellgence System s 98.*

URL1. (2007). *blog.peltarion.com.*