BACHELOR THESIS

Realization of the Cognitive Architecture LIDA in the Building Automation Domain

Submitted at the

Faculty of Electrical Engineering and Information Technology,
Vienna University of Technology
in partial fulfilment of the requirements for the degree of
Electrical Engineering and Information Technology

under supervision of

Dipl.-Ing. Alexander Wendt

Institute number: 384

Institute of Computer Technology

by

Albert Sune Belmonte

Matr.Nr. 1429804

Wien, 13.8.2015

Abstract

In the building automation domain the maximum comfort wants to be achieved; however energy efficiency has to be maximized, and both of them cannot be optimized at the same time. Cognitive agents allow solving multi-goal problems like this, having a background motivation acting in order to fulfill its given goal. A building automation agent that controls an air conditioning system will act according to these two goals and take the proper decisions to obtain the desired conditions. Even though there are several cognitive architectures, there is not yet a framework for a building automation cognitive agent. In this thesis, the LIDA architecture will be tested in a building automation problem in order to say whether it is viable for this purpose. Here, it is shown a basic implementation of a LIDA-based cognitive agent in a building automation problem. Some experiments have been made to show the results on different scenarios and prove the viability of LIDA in the building automation domain. It is also shown that some of the characteristics of LIDA complement well with the building automation problems. With these results a more complicated cognitive agent could be developed in the building automation domain. Despite existing other cognitive architectures that could be as viable as LIDA or even more, the good performance of it motivate the development of a more sophisticated software to show an actual implementation of LIDA in building automation.

Table of Contents

Abbreviations

LIDA: Learning Intelligent Distribution Agent

IDA: Intelligent Distribution Agent

ARS: Artificial Recognition System

AGI: Artificial General Intelligence

GUI: Graphical User Interface

US: United States

ECABA: Energy-efficient Cognitive Autonomous Building Automation

SiMA: Simulation of Mental Apparatus & Application

CCRG: Cognitive Computing Research Group

GWT: Global Workspace Theory

PAM: Perceptual Associative Memory

TEM: Transient Episodic Memory

SDM: Sparse Distributed Memory

DM: Declarative Memory

AI: Artificial Intelligence

1. Introduction

For many years the human being has developed science and technology in order to improve their quality of life. One of the ways to improve the mentioned quality of life is the air conditioning as a way to achieve comfort. Providing a comfortable environment can improve the human performance as well as improve the mood of everyone. This is an important factor since air condition becomes now a production-improvement factor, so it is essential to have the proper conditions in the working space to maximize the results.

1.1 Motivation

Besides that, renewable energies have made a step forward in the last years as the climatic change problem has become more dangerous. Nowadays, energy efficiency is an important aspect when talking about energy-intensive system such as appliances, boilers or pumps. One way to approach the energy efficiency is the design. During the design face, engineers can manage to create a product as efficient as possible; however this improvement is limited by the current technology. Another way to improve the energy efficiency is through a proper control of the system. Although humans can be taught to control the system in the proper way, sometimes there is the need of it getting controlled autonomously.

What it's normally seen in most buildings in these days is a programmable central station where the temperature of each room can be scheduled for different time slots [Lu10, p. 1-3]. This is a satisfactory solution but not the optimal, given the current technology, because it does not take into account the external factors that affect the solution and it does not have a predictive behavior. Another solution would be a cognitive agent, which is able to predict the evolution of the external factors and act consequently. An agent is a materialization on a software of a cognitive model that is able to perform the cognitive functions.

In order to materialize this agent, a cognitive model and a framework to implement it are needed. This framework needs to have certain characteristics so that it's good for the ECABA project since not all the frameworks can be used for building automation. In this case the LIDA, which stands for Learning Intelligent Distribution Agent, framework has been chosen to test its

applicability in building automation and determine whether or not it is possible choice as a framework to implement a building automation agent.

One of the main things when talking about buildings is its air conditioning; this is modifying the air conditions in order to achieve a higher comfort of its habitants or workers. In the early stages of the air conditioning field the main products were stoves and fans. However, nowadays there is a big diversity of products such as air conditioners or programmable stove regulators that achieve a higher control to the user. Following this line the next step is a cognitive agent focused on building automation. Unlike the current systems of controlling air conditioning, cognitive agents are able to predict, learn and act in advance to the future situations that may occur. They also have some long-term goals and motivations which will try to accomplish while choosing among all the possible options in any given situation. For these reasons seems logic that cognitive agents are the way to go for future development in the field of air conditioning.

Another important aspect when dealing with energy related processes is its efficiency. There are two main reasons for that to be a crucial aspect: the environmental situation and the long term cost reduction. Currently the environment is in a delicate spot due to the high natural resource expense of the human being and one of these resources is the energy. Whenever something is build, designed or manufactured its energy cost is evaluated in order to see how friendly it is with the environment. So in this aspect energy consumption should be minimized and energy obtaining maximized. Besides that, energy equals money in most of the cases and this means that the more efficiency the system has the more profitable will be. In this sense, a system that is able to reduce the maintenance cost of an air conditioning installation is amortized in a mid- or long-term scenario. This makes it economically sustainable which is a clue factor in the engineering field.

Overall, there is room for a performance improvement in the air conditioning field through control software's. Indeed, it is desired that it increases the performance of the current systems of control while having in mind that energy wants to be saved. It also should be able to adapt to different scenarios, because it can be performed in different external conditions depending on the localization of the building or how changing are they, and capable to predict future scenarios through evidences. With these requirements it can be thought of a building automation cognitive agent that controls an air conditioning system. In order to be able to implement it, a framework is needed.

1.2 Problem Statement

The people on the project ECABA has been working on the development of a cognitive model that fully adapts to the field of cognitive building automation. However, there is still no such a

framework to implement the model. Different types of architectures should be tried and evaluated in order to see its applicability and to decide which the best option is. In this case the LIDA framework will be tested and evaluated its viability through its application in a basic decision making problem. Between all the possibilities existing, LIDA has been chosen due to its characteristics that lead to the thought that it could be viable. The goal is to find out if LIDA can solve the tasks defined in project ECABA (Section 2.7).

The LIDA model is an extension of the IDA model with the feature that it is capable of learning from its experiences. For many years IDA has been used by the US Navy to automatize the decision making process for assigning sailors their new posts [FPJ06, p. 2]. Based on the capability of its predecessor, LIDA may be a good option for solving decision making problems. Furthermore, the possibility of learning that implements LIDA makes it more robust to possible changes or new situations that might come up and increases its autonomy.

Due to the complexity of AI software's, it's a good idea to implement the agent on an existing framework such as LIDA-framework. It provides a solid skeleton with the generic functionality of LIDA but with the possibility to customize it to the building automation application. Since it has been proved that IDA has got good results in similar situations, the LIDA-framework has the potential characteristics to become a good framework for this proposal.

In order to evaluate the viability of the LIDA, a basic software using the LIDA framework will be designed. The system to control is placed in a home scenario and it will be only disposed for domestic usages. The problem is given by the following situation:

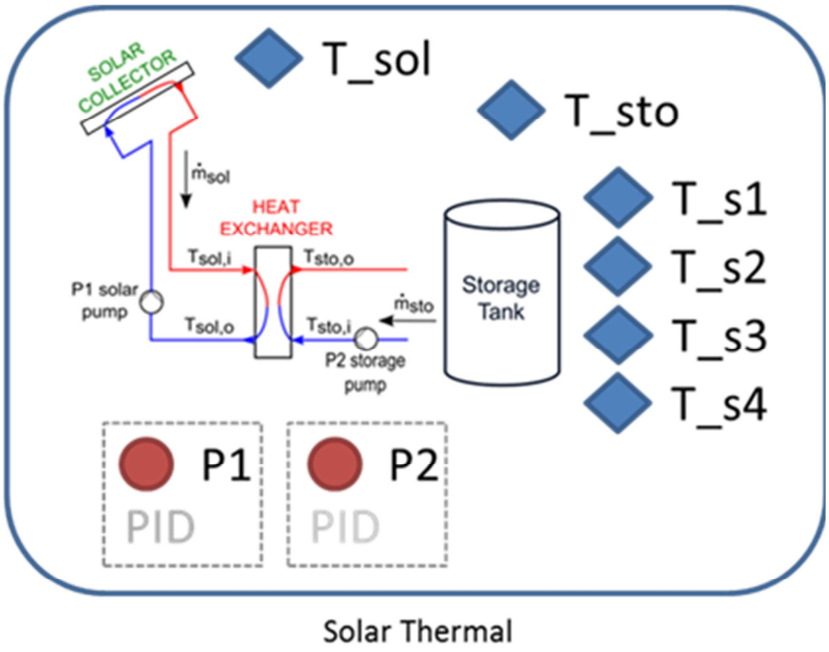

Fig 1.1: Building automation system

There is a solar panel which collects energy from the sun and transfers it to the water. In the other side there is a storage tank that stores the heated water so that it can be used for domestic purposes such as a hot shower or cleaning the dishes.

As it can be seen in Fig 1.1 there are two different water circuits. The one on the left (Circuit 1) is a closed circuit which water is used to heat up the water of the one on the right (Circuit 2). The circuit 2 contains the water from the storage tank, so when the storage water is used it must be refilled with cool water again which makes the average temperature of the tank decrease.

In order to make the energy exchange possible, a pump is located on each circuit. The power of these pumps will determine the mass flow and in consequence the amount of heat that is exchanged between both circuits. This energy exchange happens in the heat exchanger which provides the proper conditions so that the energy loses are as low as possible. On top of that the system is also provided with temperature sensors to help to take the correct decisions. There are sensors on both the solar panel and the storage tank as well as in all the inputs and outputs of the heat exchanger.

The task is to design a software implemented on the LIDA framework such that is able to properly regulate the system to satisfy the demand of hot water while minimizing the energy expense. This must be achieved regulating the pumps power depending on the external factors. What the system knows are the temperatures given by the sensors, so based on these temperatures it has to make the right decisions.

1.3 Methodology

1. In first place, the LIDA-framework shall be learned and analyzed. It is important to understand how both the LIDA model and the LIDA framework work. A deep study into LIDA model shall be done before starting to learn the framework. Once the model is understood, the framework characteristics, classes, methods and functions will be explored. It is important to have a good understanding of the framework before starting to work on it; otherwise it might lead to conceptual mistakes during the implementation. Once LIDA is understood it will be time to start the programming and development phases.
2. The next step will be to implement the domain specific functions and features to give the framework capability to define its current situation. This consists on implementing all the internal structure for the decision making process of LIDA. For this reason, new functions, classes or methods will be developed specifically for the purpose of building automation.

3. After the development of the new characteristics that allow the framework to take the decisions in the building automation domain, new implementations need to be done in order to ensure a proper user experience of the software. This consists of the GUI that will help in the interaction user-agent and will speed up the time spend on configuring the system; the implementation of simulation files in order to preset the configuration through files and to do a data post-processing; and simulation functions to achieve a likely reproduction of a real system.
4. With the development phase finished, some testing will be performed to check the viability of the agent. This will consists in a series of experiments design to test and compare the response of the agent in different scenarios as well as to see its potential in a more complicated application. These tests will give the information to conclude whether LIDA could be viable for the building automation purpose or other frameworks should be found.
5. Finally it will be discussed if the information obtained leads to think that LIDA has the proper characteristics to implement a building automation cognitive agent and why. The frameworks strengths and weaknesses will be exposed and valued in order to obtain a conclusion.

2. State of the Art

Before starting the work on a LIDA cognitive agent, it is important to get information about LIDA: how it works, what it is its structure, what has it been used for and an overview in general of the LIDA model. This section exposes an explanation about the most important parts of LIDA such as Global Workspace Theory (GWT), the learning system and the cognitive cycle. It also introduces LIDA's antecessor IDA, the architecture SiMA and explains what the project ECABA is.

2.1 LIDA

LIDA is a cognitive architecture developed by the Cognitive Computing Research Group (CCRG) of the University of Memphis. It is intended to model the way that the human performs the decision making process. LIDA is the more developed version of its antecessor IDA. IDA is a software used by the US Navy in order to perform job assignments for their sailors [FPJ06, p. 2]. The LIDA model is a comprehensible, conceptual and computational model covering a large portion of human cognition which is based primarily on GWT. The LIDA framework is based on the model and both of them are grounded in the cognitive cycle [SMF11, p. 135].

2.2 Global Workspace Theory

Global Workspace Theory is a theory of consciousness which, at the moment, is the most approved and considered. It implements a characterization of the algorithms, taking into account brain and psychological information [BF09, p. 1]. LIDA provides a working conceptual and computational model of cognition [RBF06, p. 1] that models the human cognition in the sense of GWT [Ram08, p. 1]. It also grants an accurate application of the GWT, in the sense of working on human assignments, like assigning jobs to the sailors of the US Navy [BF09, p. 1].

The GWT attempts to integrate in a single conceptual framework the role of consciousness in human cognition [FRD+07, p. 1]. Consciousness, as seen in the GWT sense, allows to deal with new or problematic situations that cannot be dealt, or at least not efficiently, by unconscious

processes. This is done by using the resources otherwise spent in unconscious processes in solving the conscious situation.

In the GWT sense, the human mind is represented by a multitude of small and special purpose processors which normally are not conscious. When these processors collide, a collection is formed. These collections perform together in order to achieve a certain task. These coalitions mostly execute ordinary activities, pursuing motor, sensory, or other tasks. A coalition that gains consciousness can broadcast a message to all the unconscious processors in order to get new components to be able to deal with an unknown, novel or complicated situation or solving a problem [FRD+07, p. 1].

In LIDA, the processors of the GWT are indeed the so-called codelets. These codelets are a few lines of executable code that perform a special purpose, active process [FPJ06, p. 2]. There are different types of codelets such as attention codelets, expectation codelets, information codelets, intention codelets, etc. In particular, attention codelets look out for situations of interest of them and try to bring them to consciousness.

2.3 Learning

The LIDA architecture will have different learning capabilities; using feeling and emotions the systems will experiment a development phase, such us a human infant, during which it will learn different behaviors before it can act as an “adult” [FRD+07, p. 1]. These feelings and emotions are thus used by LIDA as motivators and modulators of learning. LIDA comes from Learning Intelligent Distribution Agent, and it implements three ways of learning to IDA. These new capabilities allow LIDA to differentiate between perceptual learning, episodic learning and procedural learning [FPJ06, p. 2]. Perceptual learning is the capability of learning new objects, new object categories, new relations between objects, etc. Episodic learning is the type of learning that stores the information about what, where and when in the Episodic Memory, so it acts as a memory for the past events. Finally, procedural learning consists in acknowledging the procedures for executing behaviors and storing them into the Procedural Memory, which is learning new actions or action sequences to accomplish new tasks [FPJ06, p. 2]. However, this learning is not actually implemented on the work done in this thesis since the current version does not include learning yet. It is still something to have in count since it could be implemented in the future.

2.3.1 Perceptual learning

The following chapter’s information is based on [RBF06, p. 3-4]. The perceptual learning in LIDA is implemented through the PAM, Perceptual Associative Memory. It consists of a

slipnet, the nodes of which represent primitive feature detectors, individuals, a category or a relation. These relations between objects represented can be spatial, temporal or causal relations.

An incoming stimulus is descended upon by perceptual codelets (primitive feature detectors). Each of these codelets is looking for some special feature, such as a color or an angle, or more complex features, such as a red line or a blue circle. When the codelet finds the feature of its interest, it will activate the appropriate node or nodes in the slipnet, and then activation is passed. Eventually, the slipnet will get stable. At this point, nodes with activations over a threshold together with their links are used to construct the meaning of the stimulus.

The perceptual learning in LIDA only occurs through consciousness, so in order for something to be learned it has to come to consciousness before. There are two forms of this kind of learning: the weakening or strengthening of the base-level activation of existing nodes or the creation of new nodes and links.

When a new individual item comes to consciousness, a new node is created together with the links to it from its feature detectors. This new item gets to consciousness due to some new-item attention codelet. If this attention codelet accomplishes to bring the new item into consciousness, the perceptual learning mechanism created a node for it in PAM. In the same way, new categories may be created if a similarity-attention codelet notices in long-term working memory two items with considerable coinciding characteristics, and succeeds to bring these coincidences into consciousness. Then the perceptual learning mechanism creates a new category which has links called “is a” from each item into the category that has just been created. New relation nodes are created when codelets, such as relation, noting or attention ones, bring the new relations to consciousness successfully. This behavior is similar to the humans when a relation “pops into mind” or “occurs to me”.

One problem that can be thought of in this way of learning is that all these nodes and links, that sometimes are created the rate of several per second, might get computationally intractable. This problem is solved by the rapid decay of almost all the new nodes and links. In the LIDA model, perceptual learning generates trial nodes and links, and rapidly decays those that don’t quickly prove to be useful.

2.3.2 Episodic Learning

The following chapter’s information is based on [RBF06, p. 4-5]. Memory in the LIDA model interacts with conscious events for its normal functioning. Episodic learning is about interpreting the contents of consciousness in order to be able to store what has been done on each cognitive cycle, together with its place and time, into the episodic memory. In LIDA there

are two types of episodic memories: transient episodic memory (TEM), which has a small capacity and only stores detailed sensory information during a relatively small amount of time, and declarative memory (DM), which has greater capacity and is used for long term storage of lifelong events and facts.

Episodic learning results from events taken from the contents of consciousness being stored in TEM, a modification of a sparse distributed memory (SDM). The main modification consists of adding a don't-care symbol "*" to the binary content space, therefore it becomes a ternary content space while it still retains the SDM binary address space.

In the slipnet, every primitive feature detector coincides to a group of dimensions of the vectors to be saved in the SDM. Perceptual symbols, which are nodes of the slipnet, that make up an event in conscious contents are tracked back through the links of the slipnet to the primitive feature detectors. Thanks to the dimensions correspondence, a vector is created and recorded to the TEM, and this way episodic memory is affected every cognitive cycle. In addition, this episodic learning also stores the emotions and feelings and the operations done by the agent.

2.3.3 Procedural Learning

The following chapter's information is based on [RBF06, p. 5]. Procedural Memory in LIDA is very similar to perceptual associative memory, because objects, categories and relations in PAM are all represented by nodes. Similarly, behavior codelets, behaviors and behavior streams have the same representation in what is called a scheme. These so-called schemes consist of an action, together with its context and its result, together with a base-level activation, which estimates the probability that the result occurs when the action is taken in its context. The context of a scheme corresponds to the preconditions and the result to the post conditions. Both of these are nodes in the slipnet which are correctly related to their primitive feature detectors.

As mentioned, in the GWT, resources are recruited internally in order to deal with current conscious situations. Procedural memory receives the broadcast of the contents of consciousness and the schemes are activated by these contents in proportion to how well the contents match with the scheme's context, and how well the results of these schemes match one of the goals of the agent, which are specified by emotions and feelings. When activation passes in the net of schemes and it becomes stable, the schemes with a value over a certain limit are created as behavior codelets, behaviors, or behavior streams in the net of behaviors.

Empty schemes, those only containing a primitive action with no context and results, lie in the periphery of the scheme net. Each scheme has two types of activation: base-level activation and current activation. The base-level activation is a measure of the reliability of the scheme, or what is the same, the likelihood of its result occurring when the actions is taken under its

preconditions. The current activation indicates how applicable a certain scheme is to the current situation. In order to start the learning procedure, the behavior network has to choose one of the empty schemes to execute it. The fact that it has no context indicates that is always applicable to the current situation. Before executing the actions, a new expectation codelet is created by the behavior codelet. After the action is taken, this expectation codelet measures the changes produced in the environment as a result of the action being executed and attempts to bring the information to consciousness. If it succeeds, a new scheme is created if needed or, if one already exists, it is appropriately reinforced. Conscious information just before the action was taken becomes the context of the scheme and, equivalently, the result corresponds to the information that comes into consciousness just after the action finishes. That scheme is provided with base-level activation and it's connected to its parent empty scheme with a link.

Collections of behavior codelets form behaviors; these collections of behaviors codelets share pre- and post-conditions. There is attention codelets that notice behavior codelets being executed at almost the same time, although it could be in different cognitive cycles. These attention codelets try to bring to consciousness this information. If they succeed, a new scheme is created if there is not an existing one, if so it is reinforced by modifying its base-level activation. When the new scheme is created, the context is the union of all the contexts of the schemes firing together and the same with the result. Then this new scheme is provided with base-level activation and it's connected with links to the original schemes that it includes.

2.4 The Cognitive Cycle

Every autonomous agent must frequently and cyclically sense the environment and act on it iteratively, in what is called a cognitive cycle. In humans, it is known that the cognitive cycle happens between five and ten times per second, in a way that some cycles may overlap and some steps of one cycle happen in parallel with some steps of the next one. Now the cognitive cycle will be described in nine steps [RBF06, p. 2-3]:

1. Perception: External or internal sensory stimuli are received and interpreted by perception, producing the beginning of meaning. The emotions and feelings are paired with the corresponding objects and its relations by the PAM.
2. Percept to preconscious buffer: The percept, which includes the data obtained and its meaning, and possible relations are saved in buffers, which are preconscious, of the working memory of LIDA by perceptual codelets.
3. Local associations: With the percept received and the contents that have remained on the buffers, local correspondences are automatically recovered from TEM and DM and stored in long-term working memory.

4. Competition for consciousness: The attention codelets look at long-term working memory and try to bring to consciousness those events that are urgent or relevant.
5. Conscious broadcast: A coalition of codelets, normally an attention codelet and its corresponding information codelet containing information, obtains admittance to the global workspace and its content is transmitted globally.
6. Recruitment of resources: Relevant schemes respond to the conscious broadcast. These are typically schemes whose context is relevant to the information in the conscious broadcast.
7. Setting goal context hierarchy: The new schemes use the information that has been broadcasted, which includes emotions and feelings, to create new goal situation in the net of behaviors.
8. Action chosen: The net of behaviors selects only one behavior, among a recently created behavior stream or maybe an earlier active stream. Whenever a behavior is chosen, an expectation codelet is generated.
9. Action taken: When a behavior is executed, the result is that a behavior codelet performs a particular task. These codelets contain a minimum of one expectation codelet. Their goal is to keep track the operation, trying to take any result of it into consciousness, especially if it fails.

These nines steps describe in an accurate way the procedure that the cognitive agent follows through the cognitive cycle. Another way to describe the cognitive cycle is through three different phases. These phases are known as the perceive-interpret-act cycle through which LIDA interacts with the environment. This way to arrange the cognitive cycle procedure bears close similarity to many engineering life cycles [FPJ06, p. 3].

2.5 IDA

As mentioned before IDA is the predecessor of LIDA, so in order to figure out the possibilities that LIDA can offer it is a good idea to see what is IDA and what it is used for so far. IDA is a software used by the Navy of the United States which is autonomous and conscious and performs job assignments for sailors, with the use of an AI software created at the University of Memphis.

There were two goals for the IDA project. The first one is a science one; it consisted in creating a working model of the human cognition that could be used to propose possible responses to questions about the human intellect. The second one is more of an engineering goal. It consists of designing and creating a constructive application that could do the work of a human worker, someone who discusses with sailors who are about to end their current job assignments, in

normal English about their new jobs. The testing done shows that IDA did its job in almost the same way as a human detailer, with reasonable or identical results.

When looking for new assignments for the workers of the Navy, IDA looks at her environment and constructs meaning through its interpretation and chooses the important things among the rest. Through this process it answers the only question that there is: “What do I do next?” As a detailer, it communicates with sailors in English using common emails to understand their needs. It decides which jobs to offer to each worker, based on the preferences of the sailor, the current and future worker necessities of the Navy, the approximate cost, the best starting date, and other important criteria. It selects the job to offer to the sailor and negotiates with them over the course of several emails [FPJ06, p. 2].

2.6 SiMA

Normal paths in AI haven’t obtained artificial systems which can manage difficult assignments which are done by humans. The SiMA (Simulation of the Mental Apparatus & Applications) path creates a human cognitive architecture functional model through the usage of a top-down path. Via the use of a functional model, the SiMA project focuses on the description of actions that create performances and not build a performance model. The goal of the path is to create a wide system of intelligence similar to the human’s one that can deal with difficult and changing problems instead of with small or well-defined domains [3].

The first intention when the design of SiMA (called ARS in [SWJ14, p. 3]) started was to create an intelligent system capable of recognizing and understanding real-world situation. Human beings can actually perform this kind of recognition because they have what is called “feeling”. So what was intended was to design a model of the human psyche and thus an AGI architecture [SWJ14, p. 3].

Motivation is a top priority aspect when talking about autonomous agents and thus must be solved first. In the SiMA a motivational system for an artificial agent is developed using the human motivational system. It benefits of pieces of the psychoanalytic model of the human personality, which defines the functions of the human motivations, as an example for all those actions that are motivational. The psychoanalytic drive theory describes how the state of tension n the human psyche is represented and how it operates. A drive is represented by the psyche and the quantity of this drive is showed with the so-called “quota of affect” in the psyche. When quota of affect is discharged it produces pleasure and the accumulation of quota of affect generates unpleasure. The pleasure principle dictates that the goal of all psychic activity is to gain pleasure and avoid unpleasure. The amount of pleasure can indicate the goal achievement

in a feedback mechanism where pleasure is seen as a reward. Based on the psychoanalytic theory, the action that produces the maximal pleasure-gain over a certain period of time is selected. This is called the reality principle.

As explained, drives are clue components in SiMA and this is in accordance to psychoanalytic theory, where drives are the motivational element for human actions. Drives are initiated by drive sources. The values are simulated in the SiMA model and converted into SiMA data structure. In SiMA there are two types of drives: self-preservation and sexual drives. The first ones are always started in pairs and contain an aggressive drive accompanied by a libidinous one. The other ones symbolize internal desires and are formed by four parts (anal, oral, phallic and genital) which are separated in libidinous and aggressive part, as well. In the initialization phase, the drive aim is added using the agent's experience with similar situations. These drive aims are later transformed into the agent's goals. With the experience obtained by the agent, an object of a drive that is able to fulfill the drive is stored in the memory. Finally, the compliance is controlled with internalized rules by the defense mechanism. Each drive has to pass this defense mechanism and they can either pass it, be altered or be repressed [SDW13, p. 1-4].

2.7 Project ECABA

Building automation is a field with a strong link to human comfort. The goals of operation are complex, ambiguous and contradictory: indoor comfort, energy efficiency, high availability, low costs. For the purpose of improving the operations on buildings, there is the need of a cognitive model that is able to propose operation strategies on its own through the usage of the world's information and add elegancy [4].

The ECABA system pursues goals autonomously and adapts to changes in preconditions. In order to focus the design we will define a use case for the system. This use case is based on the following problem: Energy efficient building operation including user comfort.

The system shall operate the energy systems of an office building ensuring that user comfort is given, but at the same time maximizing the overall energy efficiency. The energy systems given are:

- Energy sources: solar thermal system for hot water, photovoltaic system for electricity, a heat pump that uses groundwater
- Storage: water storage tank
- Distribution: HVAC system providing conditioned air in offices, Thermally Activated Building System (TABS)

The system shall autonomously pursue the following goals:

- Energy efficient operation: during its operation time the energy consumption shall be minimized while the usage of renewable energy sources (such as solar thermal or wind power) shall be maximal.
- Maintain user comfort when required: during office hours the users shall be provided with comfortable conditions; when the building is not occupied energy saving measure shall be taken.
- Robustness and flexibility: disturbances or changes in the environment are regarded by the system and cause the system to adapt the operation of the energy systems.

The hydraulic schematics are given (e. g. solar thermal system feeding into storage tank, TABS heats rooms consecutively etc.), also the amount and location of sensors and actuators. Energy sources and distribution systems have their built-in closed-loop controls, that is, the SiMA system operates on energy management level, not on process level (e.g. it does not control how the heat pump operates, it only gives setpoints to the components).

2.8 Other Cognitive Architectures

In order to have a better understanding of LIDA, it is important to see how other cognitive architectures that have been prove to be successful work. For this reason, an introduction of two of the most important architectures, such as SOAR and OpenCog, is exposed bellow. It is a brief introduction of the main structure used and the general procedure for operating in SOAR and the motivations of developing an open architecture like OpenCog.

2.8.1 SOAR

The following chapter's information is based on [Lai08, p. 1-4]. SOAR is a cognitive architecture created in the Carnegie Mellon University and currently maintained at the University of Michigan. Between 1982 and 2007, it has gone through eight versions, each one more evolved and refined, although in this section it will only be discussed the traditional SOAR. It has shown that it is a generic and adaptable architecture for investigation in cognitive modeling over a various kinds of learning and behavioral experiences and, as well, to be advantageous for the creation of agents rich in knowledge that would be able to create different, intuitive performances in complicated and changing environments.

The traditional SOAR, from now on referred just as SOAR, is composed of only one long-term memory, which is defined with production rules, and only one short-term memory, which is defined with a graph structure with symbolic elements in a way that relations and properties can

represent objects. The appreciation of the agent of the current state is extracted from the agent's perception and through the recuperation of the information stored in the agent's long-term memory.

The functioning of SOAR is about pairing and executing rules, which provide a changeable definition of knowledge that depends on the context. The rules' conditions correspond to the current situation and the actions retrieving information relevant to the current situation. Normally, systems that are based on rules choose only one rule every time and a single action is executed while the others are rejected. This does not allow using additional context-dependent knowledge to influence the decision. By using operators, SOAR allows this additional knowledge to influence a decision. These operators are proposed, evaluated and applied by the rules. This way, rules act as an associative-memory so there is no need to select between them and can be fired in parallel. The definition of an operator in SOAR is distributed across multiple rules. Some of them propose operators that create a data structure representing the operator and an acceptable preference. Others classify operators and generate preferences of different kinds or administer some manifestation of the usefulness of the operator for the state at a given moment. And in last place, there are rules which execute the operator through adjustments done on the working memory which show the operator's actions.

In order to support the selection and application of operators, SOAR has a processing cycle:

1. Input: Variations to perception are treated and communicated to short-term memory.
2. Elaboration: Rules compute entailments of short-term memory.
3. Operator proposal: Rules suggest operators which are convenient to the state at the given moment.
4. Operator evaluation: Rules generate predilections that the suggested operators have to be selected for.
5. Operator selection: Generated preferences are combined and the current operator is selected.
6. Operator application: The operator's actions are executed by rules which coincide with the state and the operator structure at the moment.
7. Output: All the output orders are sent to the motor system.

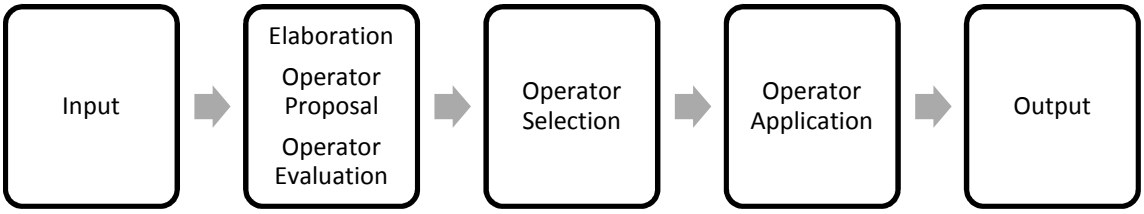

Fig 2.1: SOAR's Processing Cycle

The procedural knowledge, decision-making and subgoaling in SOAR contribute with a robust support to build it on. It has a combined short-term memory where information coming from long-term memory and perception are put together to give a united image of the state at the moment. Chunking is the learning mechanism of SOAR that converts the results of problem solving in subgoals into rules.

2.8.2 OpenCog

OpenCog is a cognitive architecture created with the capacity to unify various and differed AI methodologies together in an everyday framework operating on an everyday representation of knowledge and using an everyday arranger and I/O subsystem. Different parts of the brain contain different functions, and none of them is especially smart alone, but when they work together with the proper architecture they become in intelligence at the level of humans. At the moment, the commercial and academic information on AI has solutions to a lot of the main problems that are need to create a potent AI. When these solutions, which are partial, are put together in useful ways, the synergistic effects that come from the union of them will be able to be developed. For this reason, OpenCog aims to provide a flexible, powerful and approachable software framework that includes all the partial solutions into one unique framework. With this idea, the Open Cognition Project has been established, starting with the OpenCog Framework. This is a software framework which has been designed to promote the creation of AI systems that integrate the parts of AI structures and algorithms.

The OpenCog framework has a neutral architecture and provides utilities to create varied and rich cognitive architectures through configuration files. This is done with guiding but flexible principles, such as functional modular units. It is also done via a local AtomTable that represents knowledge and a group of MindAgent objects that implement action, cognitive or perceptual processes. These MindAgents act on this AtomTable and/or connect with the environment outside. The knowledge representation in OpenCog consists of a generalized version of the Novamente AtomTable. This technology of representation lets rival symbolic and connectionist AI systems to interact. The knowledge representation also features the usage of a language called Combo which is similar to LISP, designed especially to operate on representation of knowledge in the format of an AtomTable. As learning mechanisms, OpenCog has MOSES, a probabilistic-program-evolution module, and PLN (Probabilistic Logic Networks) module, for probabilistic logical inference. OpenCog also introduces two powerful Teaching Methodologies: AGISim and RelEx. The AGISim implementation is a 3-D simulation world with an open source that intends especially for preparation of artificial intelligence systems that are embodied. The RelEx implementation is system that comprehends natural

language, and creates logical relationships that fit for analysis through artificial intelligence algorithms from English texts [HG08, p. 1-4].

3. Design of the software

After a learning phase about the LIDA framework, the main work of this thesis, the development of a cognitive agent on the building automation domain using the LIDA framework, took place. Using the knowledge obtain about LIDA, a basic agent that controlled the system shown in section 1.3 was designed. In this section, the procedure that the agent uses in the control process will be explained. The domain specific implementation on the LIDA framework will be shown; this is the actual application of all the features of LIDA in the case of this building automation problem.

3.1 Decision making

When a cognitive agent is implemented, it is its job to take decisions to control the given system, in this case two pumps. For this reason the system has some temperature sensors that send the information to the agent. With this information the agent should be able to take a decision on the pumps' power. Here it will be explained what uses the agent to take this decisions.

When the agent knows the temperatures on every sensor it has to take the decision of whether or not turning on the pumps and, if so, at which power. In order to be capable to perform that, the agent must have some sort of rule that allows it to decide. For instance, if the solar panel has a high temperature and the storage temperature is low, then the agent should decide in favor of turning on the pumps at a high power so that the heat is transferred quickly.

This so called "rule" that the agent uses for its decision making is implemented by a drive. A drive is an indicator of how strong is some magnitude or a relation of various magnitudes. The value of the drive is compressed between 0 and 1. If it is 0 the drive does not represent a strong magnitude, however, if it is 1 the drive tells the agent that the magnitude that it's representing it's really strong. In this problem 3 drives have been implemented to model the temperatures:

- Solar Temperature Drive: when the temperature on the solar panels is high it has a value of 1 and when is low it has a value of 0. The value of the drive is given by the following expression:

$$D_{sol} = \frac{T_{sol} - T_{min}}{T_{max} - T_{min}} \qquad \textit{Equation 3.1}$$

- Storage Temperature Drive: when the temperature on the storage tank is low it has a value of 1 and when is high it has a value of 0. The value of the drive is given by the following expression:

$$D_{sto} = 1 - \frac{T_{sto} - T_{min}}{T_{max} - T_{min}} \qquad \textit{Equation 3.2}$$

- Differential Temperature Drive: when the difference between the temperature on the solar panels and the temperature on the storage tank is high it has a value of 1 and when is low it has a value of 0. The value of the drive is given by the following expression:

$$D_{dif} = \frac{T_{dif} - T_{min}}{T_{max} - T_{min}} \qquad \textit{Equation 3.3}$$

$$T_{dif} = T_{sol} - T_{sto} \qquad \textit{Equation 3.4}$$

In these expressions T_{min} is 0ºC and T_{max} 100ºC as the limiting temperatures at which water remains liquid. If a number bigger than 1 or smaller than 0 is obtained from the drive equation the value obtains the limiting value of 1 or 0 respectively.

With the implementation of this drives together with the temperature sensor values allow the agent to take the decision as it will be seen in the next section.

3.2 Nodes

The way that the LIDA agent detects what is happening in order to act is by the usage of nodes and codelets. In this section the implementation of the nodes and its linked system will be explained. The nodes are distributed basically in two different kinds: temperature nodes and drives nodes.

3.2.1 Temperature nodes

These nodes represent the state of the temperature of the temperature on the solar panel and the storage tank. There are five states of the temperature dividing the scale from 0ºC to 100ºC, as shown in Table 3.1.

Level	Minimum	Low	Medium	High	Maximum
Initial value	0	20	40	60	80
Final value	20	40	60	80	100

Table 3.1

These states are valid for both temperatures and each state for temperature is represented by a node. So there is a minimum storage temperature node and a minimum solar temperature node as well as all the other levels. There is also a parent node for each group called solar temperature node and storage temperature node that are linked to all their own levels.

A node is activated when its temperature is compressed between its limits. For instance, medium storage temperature node is activated when the storage temperature is between 40ºC and 60ºC. The temperature nodes will help the agent to take decision referred to the absolute value of these two sensors.

3.2.2 Drive nodes

Similar to the temperature nodes work the drives nodes. However the scaling of the different ranks is set different to adjust it to the behavior that the agent is desired to have. It's important to mention that all these values are fixed according to the purpose of the actions that will be defined but they can be changed easily at any moment if the situation requires so. This customization of the framework is explained in Section 4.3.

Level	Minimum	Medium	Maximum
Initial value	0.0	0.5	0.8
Final value	0.5	0.8	1.0

Table 3.2: Ranks of the solar and storage drives.

Level	Minimum	Medium	Maximum
Initial value	0.0	0.15	0.3
Final value	0.15	0.3	1.0

Table 3.3: Ranks of the differential drive.

Tables 3.2 and 3.3 show the values of the different ranks for the solar and storage drives and also for the differential drive, respectively. As it happens in the temperature nodes there is also a parent node for each of the levels of each drive. The activation of the drive nodes follow the same rules than the temperature ones.

As explained in Section 3.1, these drives will help the agent to take the decision on which actions to take. The nodes are the way that LIDA uses to know which value does the drive have and take a decision according to it.

3.3 Detectors

In order to get the nodes activated, the LIDA agent needs detectors that check how the environment is and properly activate the right node. So detector is the feature that incorporates LIDA to activate the nodes when the circumstances are the correct for that given node. Like in the node there is two kinds of detectors: the temperature feature detector and the drive feature detector. As it can be imagined by the name, the temperature feature detector detects temperatures and activates temperature nodes and so does the drive feature detector with drives. Both of them work equally, they are only different in what they detect. There are two methods featuring these detectors: the init and the detect method.

The init method, shown in Fig 3.1, calls its parent method in the BasicDetectionAlgorithm implemented by default in LIDA. Then it sets the detector parameters mode to either temperature or drive, this is made to pass this information later to the Sensory Memory in order to obtain the proper value. After that the threshold values for the specific rang on the node are obtained, for instance if the node is the high temperature node it will obtain the values 60 and 80. Finally it obtains the object that is being evaluated, solar panels or storage tank, and sets the value of the detector parameters object with the object obtained.

After that the detect method is executed. This method is very straight forward since all the work has already been made in the init method. The agent obtains the real value of the parameter on study from the Sensory Memory. This is done with the method implemented in LIDA sensoryMemory.getSensoryContent(string, params), where params contains all the information about the object in study obtained in the init method. Once the value is obtained, it is compared to the thresholds. If the value is compressed by these thresholds the node is activated by the method by returning 1.0, otherwise it returns 0.0 and the node remains or becomes not active.

```
@Override
public void init(){
        super.init();
        detectorParams.put("mode","drive");
        soughtMaxDrive = (Double) getParam("maxd", 1.0);
        soughtMinDrive = (Double) getParam("mind", 0.0);
        object = (String) getParam("object", "");
        detectorParams.put("object", object);

}

@Override
public double detect() {
        //ButtonEnvironmentPanel environment = new ButtonEnvironmentPanel();
        //double drive = environment.getDrive(object);

        double drive = (Double) sensoryMemory.getSensoryContent("drive",detectorParams);

        if(drive<=soughtMaxDrive && drive>soughtMinDrive){
            return 1.0;
        }

        return 0.0;
}
```

Fig 3.1: Coding of a detector

3.4 Codelets

Once the nodes are created and their respective detectors implemented, LIDA needs a tool to decide which node is the most important one and what should it pay attention to. This tool is what is called attention codelet. As explained, an attention codelet is a kind of codelet that tries to focus the attention on some specific information [FPJ06, p. 1]. In this case, the attention codelets used try to bring to consciousness some specific node activation. In particular, the drive differential nodes have each of them an attention codelet. This way when they obtain activation, the agent focuses the attention on them and decides for the corresponding action.

3.5 Actions

Depending on the environment at every moment the agent shall decide to take one action or another. In order to allow the agent to make this decision, the desired actions and the situations for them to be triggered need to be implemented. This implementation is done through the lidaagent.xml file which, as explained on its own section, allows to custom the behavior of the agent to the wish of the consumer.

In the created software 3 basic actions have been programmed, as it can be seen in Fig 3.2. These actions consist on turning on the pumps and setting its power to the correct level according to the environment conditions obtained through the sensors. These 3 actions correspond to 3 levels of power of the pumps. Note that these 3 levels are totally arbitrary and are defined just for the purpose of showing the agent's behavior.

The way that the actions are triggered is through the nodes and its corresponding attention codelets. So when a given noted has the attention of the agent then it looks to its actions and decides to do the one (if there is one) that satisfies the other conditions and that has the highest preference. An example of the way of "thinking" that uses LIDA would be the following: there is a high temperature in the solar panel; this attracts the agent's attention because there is an attention codelet for the maximum solar panel temperature; then LIDA looks at the actions for that node (in this case there is only one because is a simple action agent) and finally decides for the most preferable action (the only one).

This whole way of acting is implemented in the Procedural Memory on the lidaagent.xml file. There every possible outcome is defined in a scheme. Each scheme receives a name, the nodes that trigger it, the action that will be triggered and the preference that it has. Once one scheme is selected by the agent, its action is passed to Sensory Motor Memory. There the information needed is passed to the Environment for each action. In this case it just passes the name of the

action from the Sensory Motor Memory to the Environment where it will be executed. In the Environment there is a method named processAction(Object o) that receives the information about the action in the object o and executes it. In the program the object o is a string containing the name of the action which will be executed. Every action is properly programmed so that its simulation is executed.

```
@Override
public void processAction(Object o) {
    String action = (String) o;

    if ("algorithm.pumpsmax".equalsIgnoreCase(action)) {
        logger.log(Level.INFO, "Turning on pumps... MAX power!", TaskManager.getCurrentTick());
        P1="3";
        P2="3";
        newHour(3.0);
    }else if("algorithm.pumpsmed".equalsIgnoreCase(action)) {
        logger.log(Level.INFO, "Turning on pumps... Medium power", TaskManager.getCurrentTick());
        P1="2";
        P2="2";
        newHour(2.0);
    }else if("algorithm.pumpsmin".equalsIgnoreCase(action)) {
        logger.log(Level.INFO, "Turning on pumps... Min power...", TaskManager.getCurrentTick());
        P1="1";
        P2="1";
        newHour(1.0);
    }else if("algorithm.stoppumps".equalsIgnoreCase(action)) {
        logger.log(Level.INFO, "Turning off pumps... Freeze danger...", TaskManager.getCurrentTick());
        JOptionPane.showMessageDialog(null, "Freeze!!!");
        P1="0";
        P2="0";
        newHour(1.0);
    }
}
```

Fig 3.2: Coding of the actions

4. Implementation

Once the design of the agent is done, it actually needs some implementations for it to work. In this case, a GUI was needed to allow an easy usage of the agent. Since the system runs in a simulation mode, because it is not yet implemented in an actual system, a simulation system had to be designed and implemented into the agent. Also a discussion of the .xml files that LIDA uses for the implementation of all the nodes, codelets and action selection is needed.

4.1 Graphical User Interface

The first opinion that a user has about a software comes out from how it looks like. If it has a nice configuration and the different options are distributed properly, it will be more appealing for the user and will have a better response to it. Although this is not a commercial software, it's still important to keep the GUI polite, so that it's easy to be used by any user without big problems.

4.1.1 Screen composition

The framework main screen has various sections that need to be differentiated and accomplish different goals for the program.

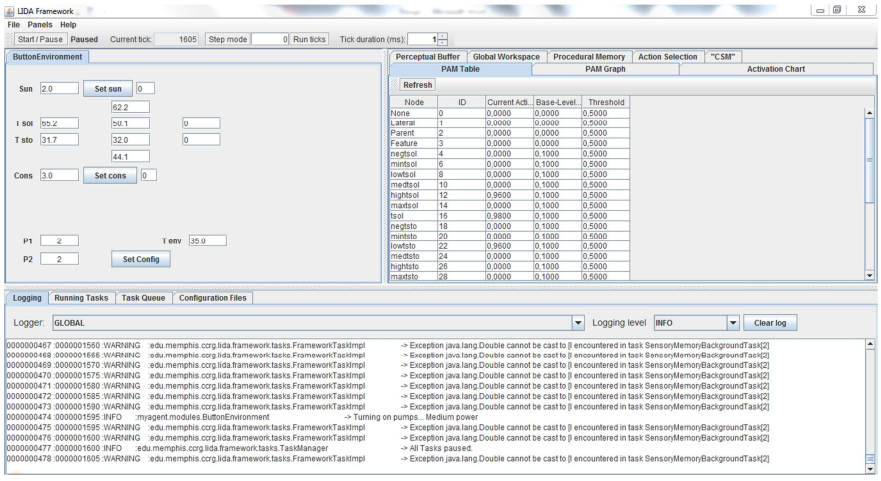

Fig 4.1: Complete GUI

On the upper side of the screen, the menu bar is located. It allows the user to start and stop the run as well as to see the current tick, activate/deactivate the step mode and set the tick duration.

At the bottom part there is various sections among which it is able to switch easily. They show internal message, task information and other important information about what is happening internally in the task execution. At the screen's center there are two different panels. The right one shows the internal node structure, its activation, the attention codelets implemented and the action selection process. This is a really important part of the framework since it shows the procedure that the agent does to select the action at any time.

Finally on the left side, the environment window is placed. This is the main window that the user will utilize to interact with the software. It is a very practical display that shows the temperatures of the different sensors and its evolution over time as the program runs and also it displays the pump power at any moment. In addition to that, there are three external indicators which are the sun, the consumption and the environmental temperature. The last of these is fixed by the user and it determines the heat gains or losses through the pipes. The sun and the consumption symbolize the ratio at which the heat energy is obtained or lost by the system. The sun value represents how powerfully the sun shines and affects the temperature of the solar panels incrementing it proportionally to its value in each iteration; in the same way, the consumption value decreases the temperature of the storage tank representing that hot water has been used in the house. These two last parameters can be adjusted "on time" as the program is running. To do that the interface is equipped with the toggle buttons "set sun" and "set cons" which when activated lock the respective value to the number on the righter box.

4.1.2 Starting interface

For the purpose of setting all this values explained above at the start of the program, the software implements a starting interface. As said, this interface allows configuring the starting point as well as how it is introduced by the user.

Fig 4.2: First window of the starting interface

The mentioned interface consists of the two input dialogs shown in Figs 4.2 and 4.3 which allow the consumer to interact with the framework. The first one, on Fig 4.2, lets to choose among three different modes of setting the initial temperatures (solar panel, storage tank and

environment). The default mode sets the temperatures to predetermined values located in a file named “defaultTemps.txt”, the file mode lets the user chose a file where the initial temperatures are already written in the correct format and the manual format leads to another three dialogs asking for the mentioned temperatures. In Fig 4.3 the second part of the interface is displayed. There the operator chooses again among the three same modes but this time it is for the simulation profile. This is the way that the sun and consumption evolve during the simulation. The two first modes work in the same way than the previous one (with its own format) but the manual mode does not ask for the evolution of these values since it would be too long to add. Instead, it sets the values to 0 by default and allows the user to change them using the “set” buttons explained above.

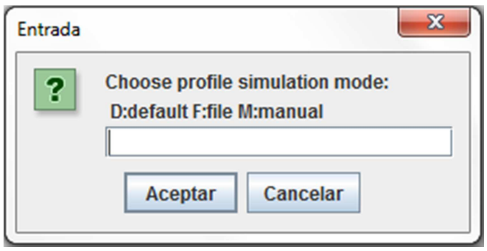

Fig 4.3: Final window of the starting interface

As it can be seen on Fig 4.1, there is also a button tagged as “set config”. What this button does is allow reconfiguring the values introduced on the starting interface. When pressed, the button recalls the starting interface so that consumer can introduce the values again. Note that this action does not stop the execution of the program, so it might be a good idea to stop it through the button “Start/Pause” in the top menu in order to implement all the changes at the same time.

4.1.3 Output File Generator

The last feature that the GUI implements is an output file generator that helps in the post processing of the data obtained. This consists of a file that the program generates automatically at an internally predetermined location of the executing PC containing all the temperatures data generated on the latest simulation run. Fig 4.4 shows an example of a short simulation data written in the output file. As it can be seen, the information is organized in eight columns each of which contains a different temperature plus the first one that indicates the iteration number “Hour x”. This is a very practical implementation since it allows a quick data treatment using “Microsoft Excel” or any similar program.

Hour	Solar Panel	Storage Tank	Solar Input	Solar Output	Storage Input	Storage Output	Sun Power	Cons Power	Pumps Power
Hour0	30	20	30.5	0	21.5	0	0	0.9	1
Hour1	27.3	17.3	28.1	24.5	19.1	22.7	0	0.3	1
Hour2	27	17.5	27.8	24.4	19.3	22.7	0	0	1
Hour3	26.8	18	27.6	24.5	19.7	22.9	0	0	1
Hour4	26.6	18.4	27.4	24.5	20.1	23	0	0	1
Hour5	26.4	18.8	27.3	24.5	20.4	23.2	1.39	0	1
Hour6	27.6	19.2	28.3	25.3	20.8	23.8	3.58	4.5	1
Hour7	31	15.1	31.4	25.7	17.1	22.8	4.77	6.9	2
Hour8	34.9	9.5	34.9	25.8	12.1	21.2	5.44	7.2	2
Hour9	38.8	4.3	38.4	26	7.4	19.8	5.83	6	3
Hour10	41.7	1.9	41	26.7	5.2	19.5	6.06	3	3
Hour11	44.3	3	43.4	28.5	6.2	21.1	6.18	2.4	3
Hour12	46.8	4.8	45.6	30.5	7.8	22.9	6.22	1.8	3

Fig 4.4: Example of an output file

4.2 Simulation

One of the clue aspects of the whole system is the simulation algorithm. The framework has been design to be implemented in a real-world scenario; however, before it starts working, some testing has to be done. That is where the simulation comes into play allowing the program to operate without the need of the real component. In this case all the components of the system have been substituted by equation that emulates the real behavior of those.

4.2.1 Sun and Consumption

The first aspect that has to be simulated is the sun. The whole system depends on the energy provided by the sun rays, so there is the need to create some artificial function that impacts in the system in a similar way that sun would. As it's explained in the GUI section there is a sun value that raises the solar panel on each iteration. This is similar to the performance of the sun, since it increases the solar panels temperature depending on the strength it shines.

Similar to this simulation, there is the consumption simulation which plays the role of the people in the house spending the hot water from the storage tank. As water is spent the tank is refilled with cool water that lowers its average temperature. The ratio at which the temperature of the tank decreases depends once again on the consumption value. Each time an iteration happens the temperature of the deposit is decreased by the consumption value.

4.2.2 Heat Exchanges

Besides the temperature changes that undergo due to the sun and the consumption, there is also a whole recalculation of temperatures. This recalculation plays the role of the heat exchanges that happen during the transport of the water, the heat exchange in the exchanger and mixture of hot and cool water at the end of the cycle.

Transport of water

Starting with the transport of the water and since the pipes are not perfectly isolated, there is some exchange of heat between the water and the environment. Analyzing this heat exchange process, it turns out to be a convection exchange. This phenomenon is governed by the heat transfer equation particularized on the convection case:

$$Q_{1\to 2} = \frac{\Delta T}{R} = h(T_1 - T_2) \qquad \textit{Equation 4.1}$$

As it can be seen in the equation above, the amount of heat transferred depends proportionally to the temperature differential. That leads the reasoning to think that the temperature at the end of the pipe will depend on the initial temperature and also on the temperature differential between the starting point and the environment. In this process more complicated equations are involved, but for the purpose of this simulation, a basic approximation that represents the basic relation between the temperatures is good enough. That said, the equation used to describe this heat transfer process has this form:

$$T_f = T_i + \eta(T_{env} - T_i) \qquad \textit{Equation 4.2}$$

In this equation the initial temperature is shifted by a value depending on the temperature variation modified by a factor η. This factor expresses how good the heat transference is, with a value between 0 and 1. It being 0 would indicate perfect isolation and the temperature would not change, and being 1 would indicate complete heat transfer (in the case of an infinite pipe where all the heat is transferred) and the final temperature would equal the environmental one. For the purpose of this case the parameter has been set to 0.1 which does not represent big loses but it's still noticeable. The reasoning behind this value is that the time that the water remains in the pipe is relatively low but the isolation is not supposed to be good, so this value offers a plausible result.

Heat Exchanger

The other part of heat transfer recalculations, and probably the most important one, relies on the modeling of the heat exchanger. Like it can be observed in Fig 1.1, the system on study consists of 2 inputs and 2 outputs. The inputs are the pipes coming from the solar panels and the storage

tank containing hot and cool water respectively. In the other hand, the two outputs are the returning pipes with cooled and warmed water. This process can be seen as a black box with two inputs and two outputs modeled by 2 equations with this form:

$$T_{sol,o} = f(T_{sol,i}, T_{sto,i}) \qquad \textit{Equation 4.3}$$

$$T_{sto,o} = f(T_{sol,i}, T_{sto,i}) \qquad \textit{Equation 4.4}$$

The equations above show that the outputs depend only on the inputs. With a similar reasoning to the one with the pipes these equation become the following:

$$T_{sol,o} = T_{sol,i} + \eta(T_x - T_{sol,i}) \qquad \textit{Equation 4.5}$$

$$T_{sto,o} = T_{sto,i} + \eta(T_x - T_{sto,i}) \qquad \textit{Equation 4.6}$$

However, this case is not exactly the same so some further reasoning needs to be made, although this is a good starting point and the final equation will have a similar form. The first thing to think about it the amount of heat that is transferred. In the previous process that depended on the temperature differential between the starting point and the environment. In this case there is no such an environment, nonetheless the inside of the exchanger can be thought of as an environment since there is where the heat is transferred. That said, the temperature of this "environment" must be calculated. This is not an easy calculation but it can be simplified into taking the mean between the two incoming temperatures.

$$T_{avg} = \frac{T_{sol,i} + T_{sto,i}}{2} \qquad \textit{Equation 4.7}$$

With this average temperature and the ratio between mass flows, an approximation of the maximum temperature variation can be defined.

$$T_{dif,max} = \frac{P_{sol}}{P_{sto}}(T_{sol,i} - T_{avg}) \qquad \textit{Equation 4.8}$$

Where P represents the power of the pumps on each circuit, which is proportional to the mass flow. Using this maximum differential and multiplying it by a reducing factor representing the efficiency in which the energy is transferred, the following final equations are obtained:

$$T_{sol,o} = T_{sol,i} - \eta T_{dif,max} \qquad \textit{Equation 4.9}$$

$$T_{sto,o} = T_{sto,i} + \eta T_{dif,max} \qquad \textit{Equation 4.10}$$

This two expressions follow the criteria of the black box explained before, where the output temperatures are obtained from the input temperatures. Note that even though in the expression the crossed input temperatures do not appear, $T_{dif,max}$ depends on both input temperatures. The value of that efficiency factor has been set to 0.8 as a reference value but it can be changed to test different exchanger performances.

End-of-Cycle Mixture

The final simulation related to heat exchange procedures is the mixture at the end of the cycle. This means when the water going out from the heat exchanger enters the solar panels and the storage tank. The mixture needs to be modeled in a way that represents that a small amount of liquid is introduced in a larger container at a different temperature. Using a decaling factor to represent the lesser quantity of water introduced respect to the one in the container both temperatures are added. After that the sum needs to be divided by the total mass of water after the introduction. Since the water in the container has been supposed as 1, the dividing factor is 1 plus the decaling factor. This procedure is shown by the following expressions:

$$T_{container,f} = \frac{T_{container,i} + \alpha T_{in}}{1+\alpha} \qquad \textit{Equation 4.11}$$

In order to determine the parameter α, a proportional law has been used. The factor is proportional to the ratio between the pipe size and the container capacity and also to the pump power, since the faster the water flows the more water enters the container.

4.2.3 Simulation Profiles

Another important implementation is the simulation profiles characteristic. This is also explained a bit in the GUI sections, referring to the way how they are introduced to the system through the starting interface. Here how they are used and what they represent will be explained.

The whole model relies on the transference of energy from the sun to the water for domestic uses. However, the sun is not like a plug where there is continuous source of power. In this case the sun shines only during the day and sometimes does not shine as bright as desired. Also the consumption of hot water is not like the energy required by a machine, people takes a shower whenever the feel like and they probably don't want to wait until the sun shines to do it. The whole point of this is that the transference of energy does not occur instantaneously but in steps, which are not always in the proper order. The ideal situation would be that the sun shines and warms up the water, then it is transferred to the storage tank and finally is consumed. This entire combinations of event needs to be modeled over time and, as mentioned in the GUI section, it's done using sun and consumption values over time. These values may be stored in files or introduced manually using the GUI. The evolution of these values represents the different values of sun power and consumption during a period of time. This is a very useful tool since real-life situations can be tested easily in order to evaluate the response of the agent.

4.2.4 Pumps and Actions

Normally with a real-life implementation once the agent has decided which action to take (as explained in section actions) the order is sent to the executing system, in this case the pumps, and they are set to its adequate power. Since everything is simulated here, artificial pumps need to be created. The pumps are defined simply by a value that indicates its current power. So there are two values P1 and P2 defining each pump. These values are used by the other simulation processes in order to calculate the mass flow to obtain the simulation results.

This model of pumps is used by the action processor. When an action referring to a pump power change is triggered, the action processor changes the value of the pump.

This is obviously a very simple model for the pumps since it only refers to its power. But there is the possibility of modeling more complicated ones just with the addition of the class pump with all its characteristics.

4.2.5 Step-by-Step Simulation

As it has been referred in other sections, this program includes an iteration system in order to execute the simulations. This iteration system is a step-by-step simulation system in which every time that a tick happens all the values are refreshed. The purpose of this is to simulate the course of the time through the simulation.

Since there is not the possibility of making a continuous update simulation because the computer works discretely, a method to trigger steps needs to be created. The most similar to the continuous case would be a refresh every time the GUI refreshes. After trying this, it has been checked that the rate of refreshing is too fast for the computer being able to achieve all the calculations. Another option is using a synchronous signal like a clock signal established every few seconds to trigger the update. This is probably the best option since it allows configuring the sampling time according to the preferences and the requirements of the situation. However, and for design purposes, this method has not been implemented but the framework grants the possibility of its implementation at any moment without the necessity of reconfiguring anything. Instead every tick is triggered by the agent's decision, which is a slower version of the one mentioned before, but for the purpose of seeing the basic behavior of the agent is enough. Although it works well in most cases, it has some limitations that lead to think that a further implementation of a clock signal would be ideal. There are two main handicaps that this system presents. The first one is that the step duration may vary depending on the time the agent takes to decide for an action or another, which is not ideal, but this one can be accepted on the supposition that the sampling time is much larger than the decision time of the agent. The second one is the one that limits the possibilities of the software at this moment on being able to

face more complicated problems. Since every step of the simulation is triggered by a decision of the agent, the agent needs to be able to decide for some action at any moment. This means that there cannot be gaps between decisions since if the environment falls into this gap in an iteration step the simulation will not work further; and also that if the agent stabilizes the simulation will end as well. As already mentioned these limitations, even though they might seem quite big, are fine for a basic behavior simulation.

5. Results

Once the agent is completed, some experimentation is needed to evaluate its behavior. It is the moment to see whether the behavior of the cognitive agent is the one expected. Different scenarios and situations can come out in real life, so it is important to check how it behaves in some particular cases. In these two experiments there will be a comparison between different situations focusing the attention in only one changing aspect. The first one will focus in the reaction of the agent in two different moments of the year. The second one will look at the differences on the starting temperatures.

5.1 Experiment 1: Solstices differences

In this experiment the agent will be tested with two different sun simulation profiles, one corresponding to the summer solstice (June 21st) and the other to the winter solstice (December 22nd). In order to adapt to the differences on the water consumption between summer and winter a decaling factor for the summer will be applied.

5.1.1 Objectives

The goal of this experiment is to test the agent in two opposite situations and see the differences on the behavior. This is an initial try out and it's only desired to get an overall overview of how the system works. Also it will help to check if the simulator parameters are correctly adapted, otherwise they will be readjusted to give viability to this and future experiments.

5.1.2 Data

For this experiment two sun evolutions have been obtained through a sun calculator [1] for both solstices and at the latitude of 48º N corresponding to Vienna. In Fig 5.1 the result curves for both summer and winter day are plot. These curves are obtained through an interpolation of the values every 10 minutes obtained with the calculator.

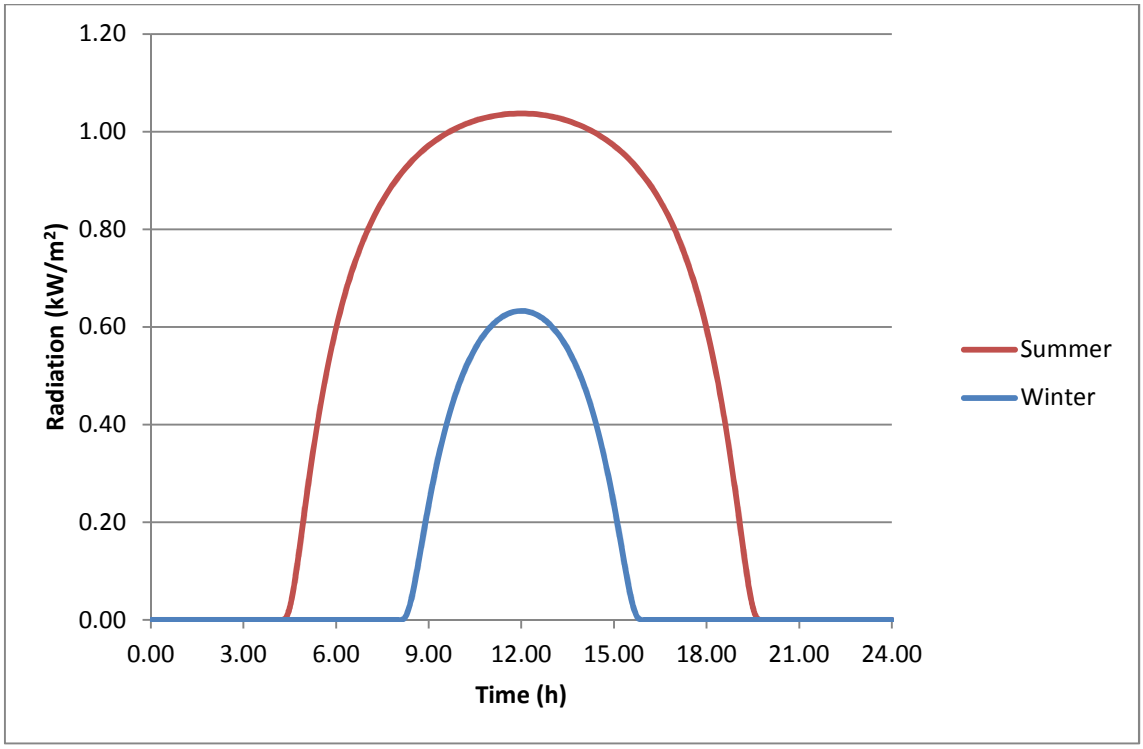

Fig 5.1: Sun power curves

However the simulation will be done with discrete intervals of 1 hour. For this reason the values used are only the ones of the o'clock hour as shown in Fig 5.2.

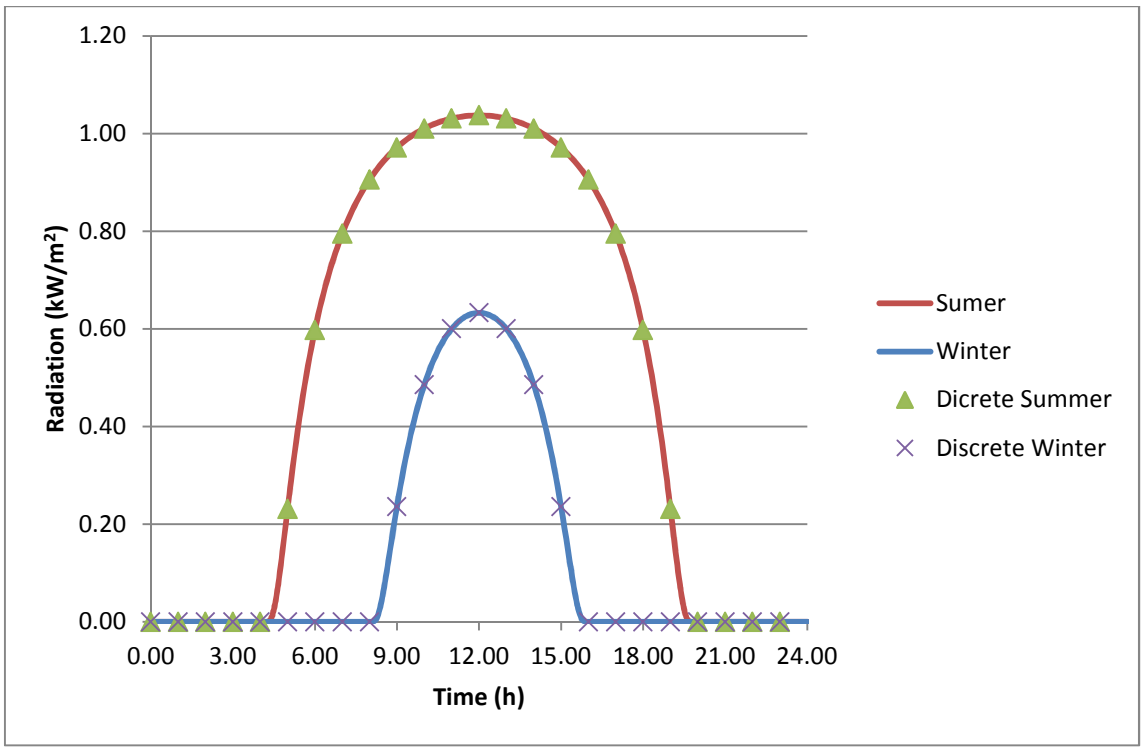

Fig 5.2: Sun power curves with discrete approximation

For the values of the hot water consumption, the results of a study done by the government of the UK [2] have been used. In Fig 5.3 the consumption of hot water throughout the course of a day is shown. In this case the result is already discrete so the actual values, properly scaled, will be used.

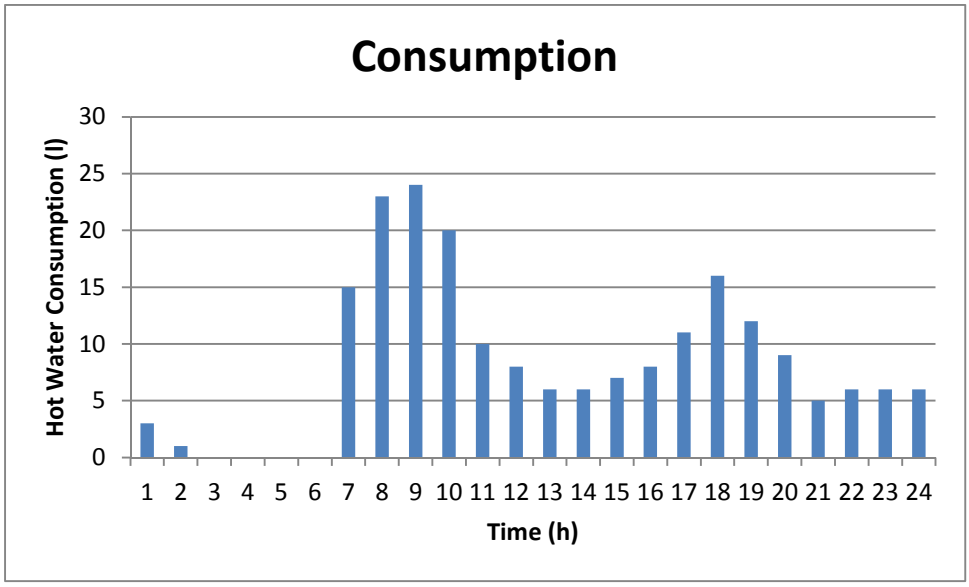

Fig 5.3: Water consumption

Using these data both simulations will be performed and the results will be analyzed. As starting temperatures, Tsol= 60ºC and Tsto=Tenv=20ºC will be set. Since the goal of this experimentation is not to discuss the about the starting temperatures but to see the evolution in stationary state this values are more or less irrelevant as long as they are in between reasonable margins.

5.1.3 Results

After performing both experiments the results can be seen on Fig 5.4 and 5.5. It is important to note that the values of sun power and consumption have been scaled to properly dimension the system. This means that since the system is not real the values are only valid for its evolution over time but not for its real magnitude. In a real case scenario the system should be dimensioned so the total power consumption matches the power obtained. However for these experiments this is supposed to be achieved.

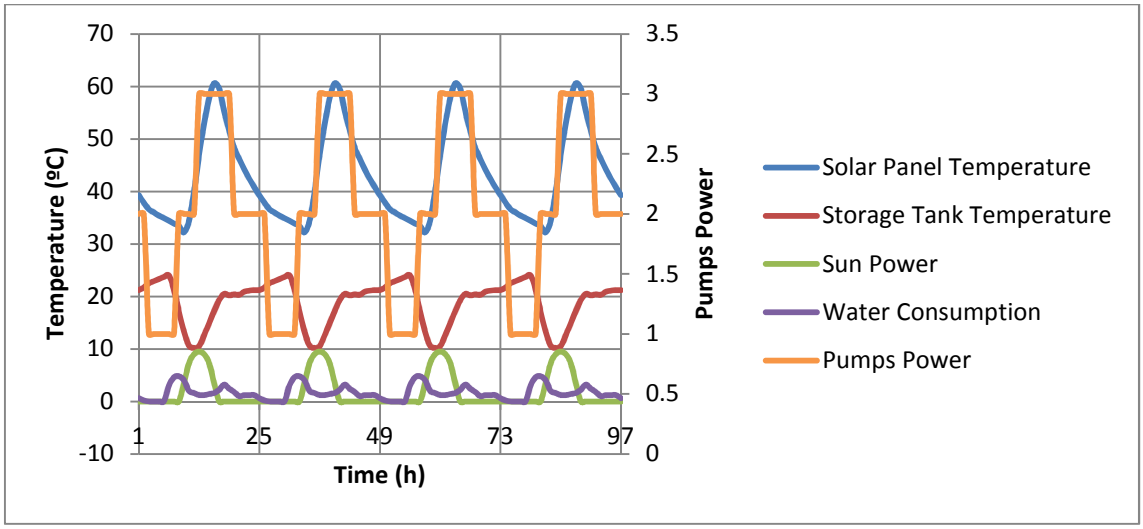

Fig 5.4: Winter Simulation

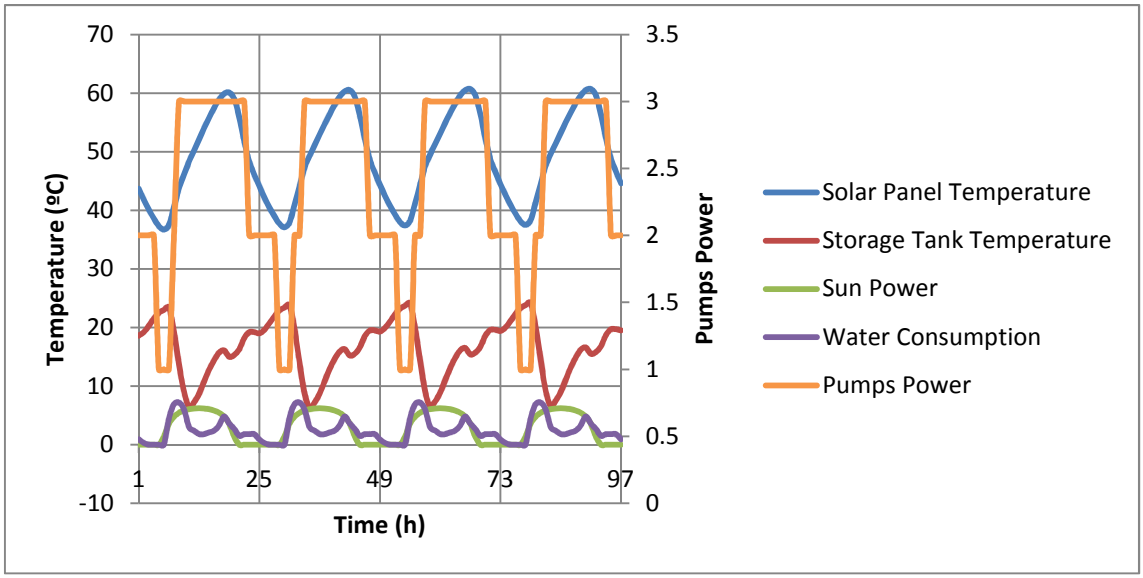

Fig 5.5: Summer Simulation

The main difference between the two results is the amount of time the sun heats up the solar panels. In winter the sun only shines for a few hours and it does not heat a lot, however in summer it shines much longer and much stronger. For these reason the agent should behave somehow different in these two cases. As it is shown in the graphs the power of the pumps does not look like the same, in winter less power needs two be transferred so the pumps rarely are at 3 power and they normally fluctuate between 2 and 1. In the other side, in summer all the power collected from the sun must be transferred to the storage tank; this has to be a quick transfer to avoid energy waste. This situation is reflected on the amount of time that the pumps are at max power, comparing it to the winter situation. In the last case the pumps remain at 3 power for a considerable amount of time trying to transfer all the heat and, when it is achieved, the power is lowered to 2 and eventually to 1 to safe energy.

5.1.4 Conclusions

After seeing the results it can be concluded that the agent behaves different in the two case scenarios. The agent recognizes when there is the need of more power, because the temperature differential is big, and when it can work just fine with a medium or low power. This happens thanks to the differential drive implemented for its decision making.

5.2 Experiment 2: Starting temperatures

For this experimentation the simulation profiles will remain constant for all the try outs; however the starting temperatures will change from one to another. With different starting temperatures the system should behave different and so should the agent. It's expected that when there is a large temperature differential the agent responds turning the pumps to maximum power but it keeps them low when there is no need of more power because the temperatures are relatively close.

5.2.1 Objectives

The purpose of this experiment is to see how different it behaves for different starting temperatures. Since, as told before, more gap between temperatures means more power of the pumps, the agent should be able to conduct the system into stationary state no matter what the starting temperatures neither the environmental temperature are. It will be useful to see if there is the need to change any parameters of the agent in order to allow controlling any initial situation.

5.2.2 Data

In order to compare the differences between different initial temperatures 4 different sets of them will be tested. The Table 5.1 shows the values of these temperatures for each test.

Test number	Solar panels Temperature	Storage tank Temperature	Environmental Temperature
1	90	10	10
2	30	20	10
3	90	10	35
4	30	20	35

Table 5.1

For this experiment the simulation profiles will be extracted from the experiment 1. Since the purpose of this experiment is not to compare simulation profiles but starting temperatures the consumption is the one shown in Fig 5.3 and the sun simulation will be set as summer (see Fig 5.2). The election of summer over winter is for the simple reason that there are more hours of sun.

5.2.3 Results

The Fig 5.6 show the results obtained through the simulation of one of the 4 cases mentioned before. The other simulations can be seen in Fig 5.7, 5.8 and 5.9.

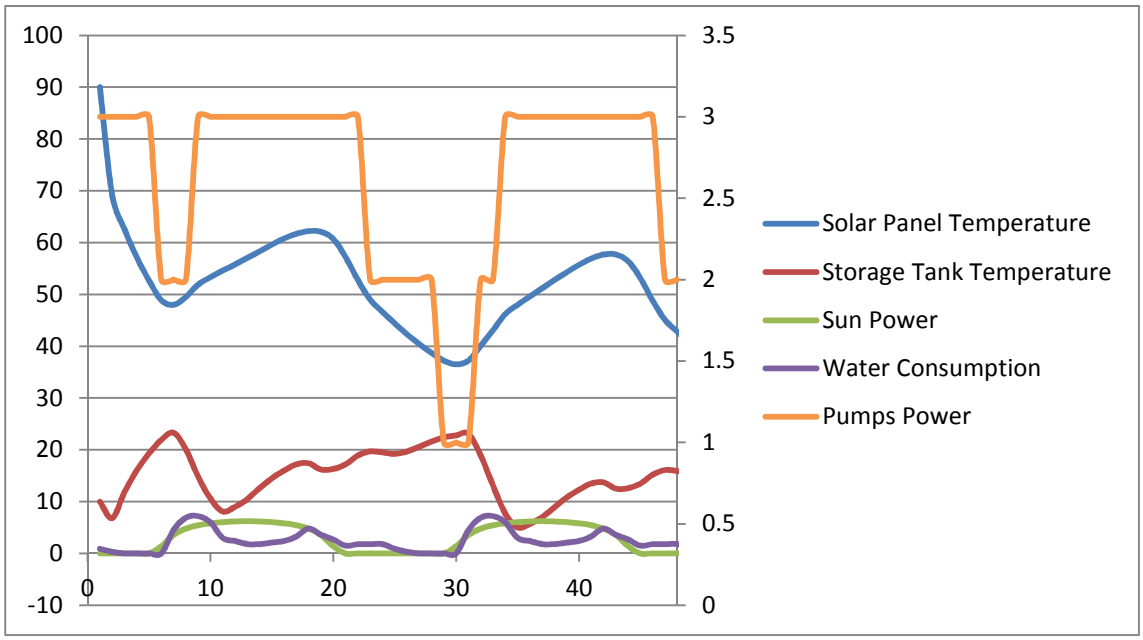

Fig 5.6: Test 1

These four experiments can be divided expressed in the following table:

	High temperature differential	Low temperature differential
High environmental temperature	Test 3	Test 1
Low environmental temperature	Test 4	Test 2

Table 5.2

In order to make a comparison between experiments, they should be grouped in pairs with common characteristics. So it will be compared high and low temperature differential and high and low environmental temperature.

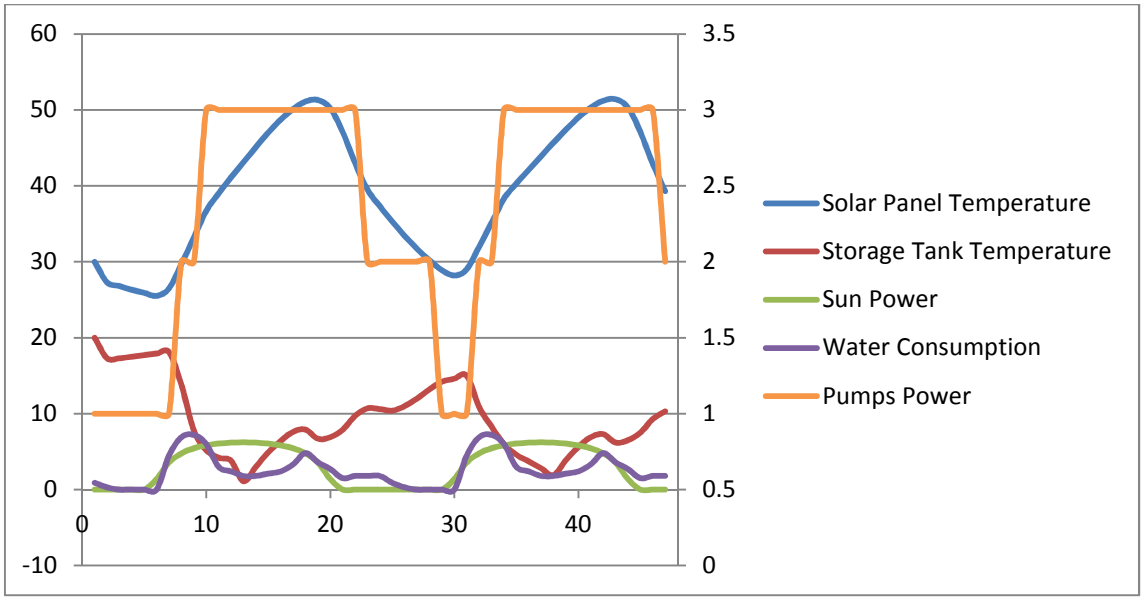

Fig 5.7: Test 2

In the first case, it can be seen that the difference relies on the initial performance of the agent, after the first 24 hours the response is equivalent in both cases. So in the case of a high differential the agent response to this phenomenon with the pumps set at max power to extract all the energy on the solar panel and make it easier to be warmed up again. On the other side, with close temperatures the agent chooses to reduce the pumps power because it wouldn't be profitable. However the agent is able to lead the system to the same path after its initial actuations.

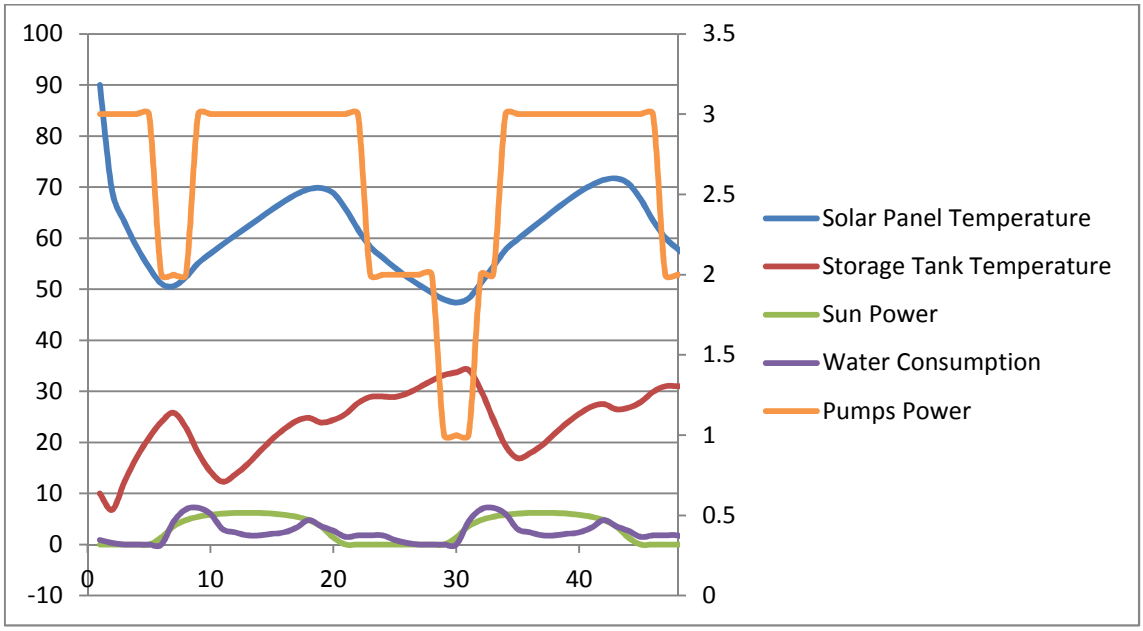

Fig 5.8: Test 3

In the second comparison the differences rely on the tendency of temperatures evolution. While at some external temperatures the system can stabilize, when this is changed the system tends to either globally warm up or cool down. The warming up case leads to think that the consumption could be higher for the amount of energy that the system is able to collect. The opposite reasoning can be done with the cool down problem, where the system is not capable to obtain enough energy to satisfy the hot water necessities. This obviously cannot be controlled since it depends on the external situations; however the implementation of a predictive motivation that incentives to store hot water when low temperatures are expected could be studied.

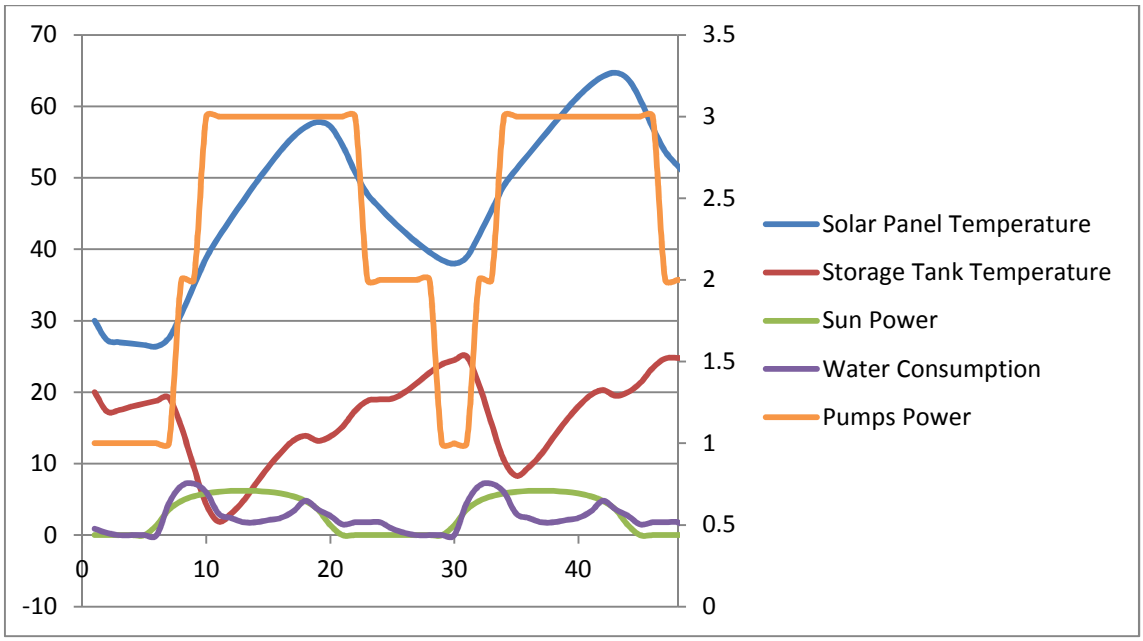

Fig 5.9: Test 4

5.2.4 Conclusions

From this experimentation it can be conclude that the system has different responses for different initial temperature differentials, however all of them lead to the same stationary result since the system is capable to neglect the initial situation in the long run (+24 hours). It also has shown that the external temperatures do not influence significantly the response of the agent since it is a factor that affects equivalently both temperatures and the agent's decisions are only based on the differential of temperatures. However the implementation of new decision rules that affect only one of the temperatures could lead to a difference in this result.

6. Conclusions

The main goal of this thesis was to test the viability of LIDA, and especially the LIDA framework, on the field of building automation. Now it has to be evaluated whether or not it is a viable architecture for this field of application. It also will be appropriate to see which the advantages that LIDA provides as a cognitive architecture are that make it a good choice for this purpose. As well as the advantages, the problems that come along with LIDA should be mentioned. Finally the overall results obtained for the practicum case should be evaluated.

6.1 Discussion

First of all, the question of "Is LIDA a viable architecture for building automation?" should be answered. The answer is "yes". There are various reasons for that answer. The first of them is that IDA has shown its good results in a decision making problem such as the assignment of new jobs for sailors for the U.S. Navy. There are some similarities between these two apparently quite different cases. Both of them involve a decision making process and require having a long term strategy to achieve a certain goal. In the two cases, the agent has to solve a multi-objective problem where different goals want to be accomplished but not both can be achieved at the same time completely. In the U.S. Navy case, the agent wants to make the best decision for the Navy necessities; however, it also needs to satisfy the sailors' preferences which may differ from the necessities of the Navy. In the building automation case, LIDA should try to grant maximum comfort but at the same time should reduce as much as possible the energy usage. In most cases, accomplishing the ideal comfort scenario may require more energy and the agent needs to take into account both goals to take a compromise solution that satisfies both of them. On the other side, it is yet early to say that LIDA is able to anticipate upcoming situations and act in advance to them. In some cases, the agent will have to foresee the future situation and take an action according to that new scenario even though it might not be optimal at the current moment.

Another reason that indicates that LIDA has the potential of being a proper building automation cognitive architecture is the results obtained by the software developed in this thesis. Although

it is a basic case of a building automation problem, the good outcomes obtained anticipate that in a more complicate case LIDA would be able to perform well. The agent showed that is capable of acting according to the situation in two extreme cases such as winter and summer. It acted as it was expected to do, and it can be said that a human controlling the system wouldn't had made any meaningful improvements. It also showed to be capable of redirecting to different initial scenarios into the same steady state. These results encourage more experimentation in order to take to the limit the potential of this implementation of the LIDA framework on the building automation domain. The implementation of new decision rules, new action or more complicated behaviors on this agent would allow having a better comprehension of the viability of LIDA in building automation. It also provides a motivation to develop an agent controlling a more complicated system, since there are proves of the viability of this framework in this domain.

The last reason to think that LIDA is a good architecture for the building automation domain is its potential for more complicated scenarios and its customizability option. The implementation of new feature detectors grants the possibility to track every desired characteristic. In the software developed, the agent made decisions based on the temperature of two points and the differential between these temperatures but it also kept track of other temperatures. However, new detectors could be implemented to grant information about the environment humidity, the weather forecast, the energy cost or the amount of incoming/outgoing power. Along with these detectors new nodes should be implemented to receive the corresponding activations. It also will make sense to implement new actions according to the new and more complicated environment. These new actions and the ones already existing will have new decision rules based on the new information. As an example, if the agent is capable of now tracking humidity, there should be a difference on the action taken in different humidity intervals; the temperature at which comfort is achieved at low humidity it will no longer be the same at high humidity. Since in building automation every system has different relevant information, the correct outcomes depends on whoever uses the system and the preferences or optimality may change over the course of time, the possibility of customizing or changing the decision making process makes LIDA a valid architecture for a cognitive building automation agent.

6.2 Future Work

After having evaluated the results obtained during the realization of this thesis, it's important to explain what can still be done to continue on this work. The main part of this thesis has been the development of a software using the LIDA framework that controls a pump system to regulate the temperature of a storage tank for hot water. At the moment there are only three actions that

the agent can decide between: maximum, medium or minimum power. In the future, more actions can be introduced in order to fulfill other more uncommon situations. These actions could be to prevent the system from dangerous temperatures for the system, such as risk of boiling or freezing, energy saving mode, progressive start or emergency stop.

The implementation of new drives is something to also consider. For now, there are only temperature drives, which are enough for a basic control, but the addition of new ones would allow a more accurate decision. Taking into account too many things though also requires more processing resources, which can be a problem. However, a point between the current state and the excessive point can be found.

Finally, it will also be appropriate to extend the system to control. Currently the system is a basic one with few things that have to be controlled; only the two pumps. Adding new energy systems to control would exponentially increase the complexity of the software but would make it more general an appropriate for an actual house.

Hence, there is still a lot of room for improvement and, since the results obtained with the experimentation done have been positive, it’s encouraging to keep working on the development of the agent.

Literature

[BF09] Bernard J Baars and Stan Franklin. Consciousness is computational: The lida model of global workspace theory. *International Journal of Machine Consciousness*, 1(01):23–32, 2009.

[FPJ06] Stan Franklin and FG Patterson Jr. The lida architecture: Adding new modes of learning to an intelligent, autonomous, software agent. *pat*, 703:764–1004, 2006.

[FRD+07] Stan Franklin, Uma Ramamurthy, Sidney K D'Mello, Lee McCauley, Aregahegn Negatu, Rodrigo Silva, Vivek Datla, et al. Lida: A computational model of global workspace theory and developmental learning. In *AAAI Fall Symposium on AI and Consciousness: Theoretical Foundations and Current Approaches*, pages 61–66. AAAI Press Arlington, VA, 2007.

[Ram08] Uma Ramamurthy. Might a lida controlled robot be phenomenally conscious? In *Proceedings of the Nokia Workshop on Machine Consciousness*, pages 32–33, 2008.

[RBF06] Uma Ramamurthy, Bernard J Baars, and Stan Franklin. Lida: A working model of cognition. 2006.

[SWJ14] Schaat, S., Wendt, A., Jakubec, M., Gelbard, F., Herret, L., & Dietrich, D. (2014). ARS: An AGI Agent Architecture. In *Artificial General Intelligence* (pp. 155-164). Springer International Publishing.

[SDW13] Schaat, S., Doblhammer, K., Wendt, A., Gelbard, F., Herret, L., & Bruckner, D. (2013, November). A psychoanalytically-inspired motivational and emotional system for autonomous agents. In *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE* (pp. 6648-6653). IEEE.

[Lai08] Laird, J. E. (2008). Extending the Soar cognitive architecture. *Frontiers in Artificial Intelligence and Applications*, *171*, 224.

[HG08] Hart, D., & Goertzel, B. (2008). Opencog: A software framework for integrative artificial general intelligence. *Frontiers in Artificial Intelligence and Applications*,*171*, 468.

[SMF11] Snaider, J., McCall, R., & Franklin, S. (2011). The LIDA framework as a general tool for AGI. In *Artificial General Intelligence* (pp. 133-142). Springer Berlin Heidelberg.

[Lu10] Lu, Jiakang, et al. "The smart thermostat: using occupancy sensors to save energy in homes." *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2010.

Internet References

[1] Calculator of solar insolation, http://pveducation.org/pvcdrom/properties-of-sunlight/calculation-of-solar-insolation, PVEducation, Properties of the Sunlight, Terrestrial solar radiation, accessed on 2015-06-08.

[2] Measurement of domestic hot water consumption in dwellings, https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/48188/3147-measure-domestic-hot-water-consump.pdf, accessed on 2015-06-08.

[3] SiMA description, http://sima.ict.tuwien.ac.at/description/, TU Wien, SiMA homepage, accessed on 2015-08-05.

[4] ECABA - Energy-efficient Cognitive Autonomous Building Automation, http://sima.ict.tuwien.ac.at/projects/ecaba/, TU Wien, SiMA homepage, accessed on 2015-08-05

Declaration

Hereby, I declare that this work has been completed without illegal aid from other parties and without usage of other than the mentioned resources. Data and concepts, which are indirectly assumed, are labeled with the declaration of the source.

This work has not been used in examination procedures at home or abroad neither in this nor in similar form.

Vienna, 5th July 2015

ALBERT SUNE BELMONTE

Appendix

Lidaagent.xml

One of the strengths of LIDA is the usage of .xml containing the data about the agent implementation. The most important file, the lidaagent.xml, is the one that contains all the customizable information about the agent, such as nodes, codelets and decision making values. This excludes all the functions such as detectors or actions but not the decision of whether or not they are included. Using the xml tags system, which is very friendly for those who are not used to programming languages, all the basic information about the structure of the agent is introduced.

The information stored in the file is very wide, however, and for the purpose of this agent's implementation, the most important one is the following. It contains a set of modules that let the agent know where are the files on which the environment, the sensory memory, the PAM, the attention module, the procedural memory and sensory memory among others that have a less important role in the implementation of the agent. In the PAM, the information about the nodes is written; the links, the parameter values and the corresponding detector for each node are assigned. The attention module is the one containing the attention codelets; in this module every attention codelet used by the agent is implemented. Finally, the procedural memory and the sensory motor memory carry the information about the action selection procedure.

One of the advantages of this .xml file system is that is very simple to understand and can be customized for a non LIDA expert without many problems as long as it has a basic knowledge of it. Another advantage, and possibly the most important one, is that the behavior and the decision choice of the agent can be changed according to the current necessities. So if when the agent is first programmed the building requires some rules but with the course of the time these rules change, the new behavior can be achieved through this file. Of course this requires that all the functions need to be already implemented. Since in the field of study all the possible outcomes are limited this condition is not a big deal.