Collective solving of problems from Multiagent systems: a dialogue between theory and classroom

Offray Vladimir Luna Cárdenas [*]

November 19, 2021

Original Spanish article date: October 2007

Abstract

The present article's objective is the characterization of the collective problem solving from the multiagents models perspective. From the theoretical approach, a point of view was assumed considering the russian socio-historic school, distributed cognition and the multiagent systems theory and, regarding the methodology, modelling and simulation was chosen as a way to approach to the understanding and the characterization, at the same time that a classroom experience was stablished to dialog with the model, and they, model and classroom experience, inspired and change both of them mutually. A model was obtained wich characterizes the collective problem solving in multiagent systems using two parameters of the individual agents: coherence and competence, and one of the problem: modularity, and the model was validated showing emergent phenomena as in the model as in the classroom experience.

Keywords: Collaborative Learning, Modelling, Problem Solving, Simulation, Cognition, Decentralization, Programming, Teamwork, Classroom Teaching/Practice.

Introduction

How problems are solved in collective? This is the question that this article tries to address. The appropriateness of the question is bold by the fact that the russian socio historic school and the distributed cognition have showed up the importance of environment, mediations, the others and the sociocultural in the cognitive development and also the fact that intelligence is not only "inside" the head, but also that it emerges and is given in and thanks to the collective. The preminence of the social software, in sites like Wikipedia, SourceForge and its free software projects, Amazon, Youtube, show informational explicitations of this collective intelligences, How this intelligence is given and operates, and how the collective is related with the individual, how "continuous" are stablished and other many questions are part of this relatively new field, with important theoretical challenges and overwhelming practical consequences, in particular and from the subject that concern us, in the cognitive and the educativa. This article show up an aproximation to the study of this collective intelligence and its *continuous*, especifically inquiring about the collective problem solving.

For that inquiry two talking approach has been taken: an algorithmic model with a computational implementation and a classroom associated experience. The last one has been tackled with a perspective based on the postvigostkian school, the distributed cognition and working experiences with undergraduate university students in a span of time of three semesters, using technical mediations like social software (Wikis) and graphical multimedial object oriented programming environments (Squeak

[*] *Email:* `offray@mutabit.com`; *Web:* `http://offray.tiddlyspot.com/`

- eToys) or with a strong visual component (*Bots Inc*). The first one has taken a methodological approach from the multiagent based sintetic modelling, from wich some authors (Pfeifer y Scheier, 1999 y Macal y North, 2005) have said to be a third view between the analitic and the sintetic or between inductive and deductive. This place is even more hibrid if the interactions between the formal model and the classroom experience are taken in account, and was the place where this inquiry was given, between findings, reconciliations and *aja! moments*, fruit of inspiration, but most of all, like the old sayind said, from transpiration.

First and quickly, the theoretical elements that allow the formulation of the model are shown. At this point the formal presentation is stoped to tackle the dialog between the formal model and the educative experience, that progress both in pararell and in a dialogistic way. The elements of the educative experience are considered quickly to bring pass to the complete model and finally the conclusions and recomendations are presented.

Theoretical elements

An agent is considered as an entity which is in and environment and is capable of making actions upon it in order to reach an objective or possible final state of the environment.

Regarding environments, they're considered:

- Accessibles/Inaccessibles: If for an agent all the information of the environment is available, then the environment is said accessible, otherwise is considered Inaccessible.
- Deterministic/Non-deterministic: If the same actions of the agents in the environment produce the same outputs independent of other factors, the environments will be called deterministic and in any other case they will be non-deterministic.
- Statical/Dynamical: If the environment only change for the action of an agent, it is said statical. If the changes happen besides for factors that are outside the influence of an agent, then is said that the environment is dynamical.
- Discrete/Continuous: A discrete environment is the one in wich actions and perceptios on it are a finite number.

There are various taxonomies also associated to the agents. Is said that agents can be: biological; robotical (like the ones in research and industrial applications(, computational and Biologic Cultural[1].

Wooldridge (1999) [2]. points three important characteristics in the agents which we're going to adscribe intelligent behavior:

- Reactivity: thanks to it an agent respond in a timely fashion to the changes of the environment, in order to reach its objective.
- Proactivity: it referst to the fact that the agents can take the initiativa in order to stablish their desing objectives and show a guided behavior by goals.
- Social ability: is related with the possibility of the agents of interacting with other agents.

[1]In this category are located the Dennet's criatures which are divided in: Darwinians, Skinnerians, Popperians y Greorians. This last ones, to which we belong, can use instruments like the words.

[2]Formal definitions of Environment States, Actions, Run, States Transforming Function, Environment and Agent in this section are based of Wooldridge ones, and they are the building block to arrive to our own formal definition of a Problem.

To reach the formal definition of agent, in a constructive fashion, the formal definitions of the elements involved in it will be presented.
Definition.***Environment States****:* *An environment can be in a discrete set S of instantaneous states that is simboliced by*

$$S = \{s_0, s_1, \ldots\}$$

Notice that has been chosen an environment as a discrete set, nevertheless this does not affect the possibility of our agent operating in continuous environments, without lost of generality, because we will suppose that perceptions of the agents over continuous environments, can be mapped, regarding their action, in discrete states.
Definition. ***Actions****:* *The actions that the agent can do are defined as the finite set:*

$$Ac = \{\alpha_0, \alpha_1, \ldots\}$$

We have left behind of this model, until this moment, the autonomous character of the agent and its objective, but we will take care of them in a while, to concentrate in the environment and the actions, right now
Definition. ***Run****:* *We will say that actions of agent modify the environment through runnings, which are alternated sequences of states and actions, because every action of an agent over the state of an environment, produce a new state. So a running r is:*

$$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \ldots \xrightarrow{\alpha_{u-1}} e_u$$

and, following the Wooldrige proposal, we define:

- *$\mathcal{R}$: the set of all possible runnings r*
- *$\mathcal{R}^{Ac}$: the subset of $\mathcal{R}$ which runnings end in actions.*
- *$\mathcal{R}^{S}$: the subset of $\mathcal{R}$ which runnings end in states.*

Definition. ***States transforming function****:* *A states transforming function τ, which represents the actions of an agent over an environment can be seen as:*

$$\tau : \mathcal{R}^{Ac} \to \mathcal{P}(S)$$

wich sends a subsets of runnings, that end in actions, in a subset of states, which are the results of such actions over the previous states.

Definition. ***Environment****:* *An environment Env is defined as the triplet of the form $Env = \langle S, s_0, \tau \rangle$, which S is a set of environment states, $s_0 \in S$ is the initial state and τ states transforming function. Notice that this definition emphasizes the possible states of the environment, the initial state and the function which transforms from this initial state to any of the possible states.*

Definition. ***Agent****:* *Agents which habitate the environment are defined like functions which maps from runnings that end in states to actions. That is to say the main function of an agent is tranforming states of the environment into actions. Resulting actions as well generate new actions that influence lately to the agent (this is the notion of agent - environment interaction). Formally we have:*

$$Ag : \mathcal{R}^S \to Ac$$

The fact that the function, which defines the agent, has account of running which end in states and not only the states, mean that the agents has some historic sense (with exception of the purely reactive agents, that we will mention briefly later).

However, the states of the environment are not equal to what the agent "knows" of this, because, in general, agents habitate inaccessible environments. This introduce the notion of perception. A perception is that which is grasped by the agent from the environment's states (which is not unavoidably related with real states) through a sensor function

$$\text{sensor} : S \to \text{Per}$$

and the action is a mapping which goes from the perceptions to the set of actions:

$$\text{action} : \text{Per} \to Ac$$

an agent is, then, a sensor-action pair $Ag = \langle \text{sensor}, \text{action} \rangle$.

A problem will be defined in the model as the diference between the objetive state of a multiagent environment and the actual state of it and the problem solving will be the execution of actions which have as finality make a match between the (future) objetive state of the environment and the state percibed as the actual one[3].

Definition. ***Problem.*** *In terms of the previosly stablished notations, we will say, then, that a problem P is the difference between an s_0 state and an s_f state from the environment Env. As well the s_0 state is characterized by a ordered set of values $s_0 = \{v_1, v_2, \ldots, v_n\}$ and the e_f state is characterized by a ordered set of values $s_f = \{v'_1, v'_2, \ldots, v'_n\}$. Is taid that the problem persist if there is at least one $v_i \neq v'_i$ for i between the indices of the states s_0 and s_f and is said that the problem was solved if $v_i = v'_i$ for all i in the indices of s_0 and s_f. The size of the problem is the number of values for which $v_i \neq v'_i$. That is to say a solved problem is the one which has 0 size.*

Dialogue between the model and the educative experience

The associated educative experience of the model and the model itself were developed in parallel during a time of three semesters (two in 2006 and the first one in 2007). The objective in this time was to put in dialogue both the experience in the classroom and the multiagent systems theory to get mutual inspiration. This section want to give account of the more important moments of such dialogue, identifying the more importand landmarks in the building of the model. *First moment: No competence, coherence or modularity*

In the first semester of the experience, the model was not clear (in fact only few things were clear at this time and there was not model properly), so was intended to observe the emerging of the collective problem by setting up a digital mediation which could work as a common space for the bird of such problems. So a collaborative system with memory for the creation of hipertextual documents, called Eduwiki[4], was set up and put to dispossal of the students ant teachers of the class Introduction to informatics. It was asked to students to chose a problem interesting for them to be work out during the course, and they woulb be publishing a personal diary of their advances on this project and also it was

[3]In this sense it can be said that our problem solver multiagent system is a diferential engine from the perspective of Minsky's Society of Mind.

[4]There are many wiki variants. In their original conception, wikis are just systems for quick hipertext creation, but the collective character and editing history is what made them so pertinent in knowledge management, enterprise documentatation scenarios and have creatd the success of the Wikipedia project, for example. The address of the educative wiki was `www.eduwiki.info` but not more online an is available from Internet Archive or other online memory systems.

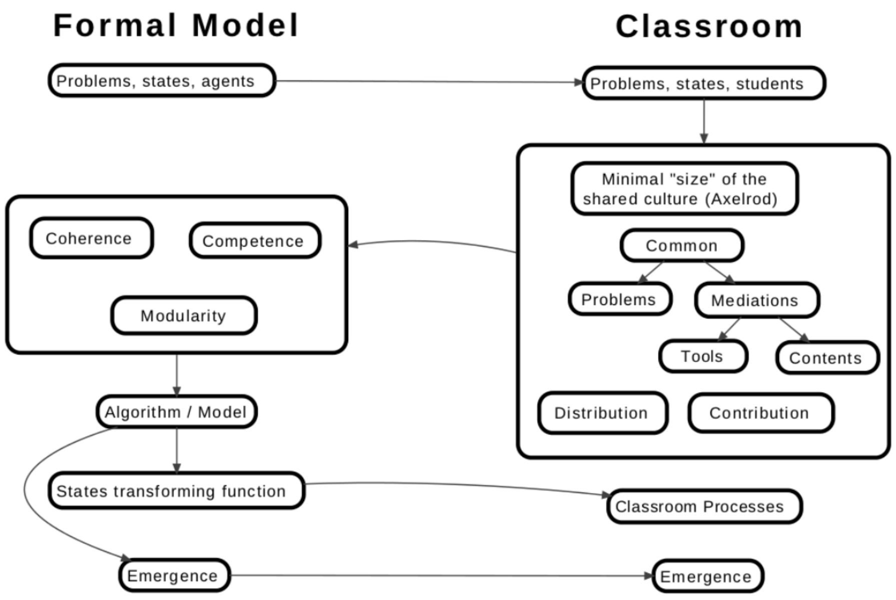


Figure 1: Schema of the dialog between the formal model and the classroom

asked to read and contribute to the personal diaries of their classmates and it was hoped that, because different problems involved the same algorithmic concepts (loops, conditionals, etc.) the collective problems on such concepts would rise, and the root of such collective problems would be the particular and different problems of every student.

This set up did not have success. Students did not read each other, nor contribute consistently to the pages of their classmates, and because of that the collective problem would not rise up. Besides of that, being introduction to informatics and elective open subscription class, we have students from different places (systems ingeneering, biology, informatics and mathematics) and from different semesters, with different concerns that were reflected in their selection of especific problems. Despite of the existent suggestions about programming languages and environments to use, with a predilection for the ones with simple syntax (Python, Smalltalk/Squeak), the fact of having some many thematic and implementation variants, besides the common practice of not reading the classmates, made extremely difficult the emerging of a collective problem. However in this scenary the first particular questions and temptative hypothesis were concived, feeded by the reading of the text *The complexity of cooperation* (Axelrod, 2004), in particular the article about culture spreading

- Because of the extremely diverse character of the *classroom microculture*, it was needed a unifiying factor for the collective problem solving and a minimal size of the shared culture, expressed in cultural practices inside the classroom, as the frecuent publishing, the lecture and contribution between pairs and the cultural mediations for programming.
- The unifiying factor was not located in the *classroom microculture* but in the shared macroculture, in this sense, the use of computers influenced profundly by the multimedial was a common reference en the young people, specially the students of this class, so an programming application with emphasis in multimedia and easiness of use was chose for everybody: Squeak
- A good portion of extra class work was related with the search of proper literature that helps in

the learning of Squeak. The times associated to preparate the experience and make a metareading of it were a limiting factor in the belonging of the teacher to the microcommunity of practice, at this was not allowing to give course to the process, *from inside*.

- The first signs of colaborative work started to show and the favorable grupal response of using a common tool, of the afore mentioned characteristics, was generaliced, however we did not reach the collective solving of problems.
- The student's evaluation instrument was modified, so it could give explicit account of the values that I want to propagate in the microculture (this could be related with the article about emerging of social rules of Axelrod in *The complexity of cooperation*, where the evaluation instrument make explicit the rule).

Second moment: coherence and competence, but not modularity

With the experience of the previous semester, Squeak was chose from the beginning of the next one as an object oriented programming environment (OOP), easy, mutimedial, which makes use of *etoys*, an easy programming drag and drop system. It was about getting first the minimal size of the shared culture to solve problems, so after covering the fist themes in the beginning of the course, the introductory part to programming was started, stablishing a minimal of common activities which, even if they can have small changes from student to student, they were using the same fundamental programming concepts of object oriented programming (instanciation, agregation, polymorfism and encapsulation) and related with the use of the interface.

The activities for the consolidation of the minimal size of the shared culture were two: the creation of an interactive book, using the facilities of the Squeak's *bookmorph* and an interactive game using *etoys*. After that we proceeded to start a collective problem, that was to create a game, which, in turn, was the solution of a collective problem: to get out a bunch of virtual bugs from a labyrinth, using the assignation of individual functionalities, differents and complemetary for each of them (there is a popular game of the 80's for game consoles called *Lemmings*, with various clones like *Pingus*; ours was, in that sense, another clone). This allowed to have two reading leves about the solving of the problem: in the level of the students, and in the level of the game itself.

Each student had to be in charge of programming a specific functionality for the bug and all of them had to be integrated in a virtual scenary where various bug exemplars (instances in the sense of the object oriented programming) would be locked. This second experience showed a lot of things referred to the interaction between the agents (students, virtual bugs) and the environment (classroom-mediations, labyrinth):

- The shared space of interactions between agents allow coordination between them. When a functionality, that was pertinent to various subproblem, emerged, this acted as an *attractor* (in the sense of the dynamical complex systems). This was saw in the wiki pages for student's publication, and also in the formulation of questions to *other students* in the classroom sesions (a *microcommunity of discourse* was in formation[5]), and also in the codes of major functionality for the virtual bugs or the environments that they inhabited. In the case of the wiki pages, the classroom talks and the code of the software objects and their environments, the *attractors* help to organize the activity of the problem solving carried by the agents, being them other students in the classroom (natural agents) or software creatures (artificial agents).
- The first cases of bifurcation, alternative solutions to the same problem, appeared. This shown a first approximation to the modularity in terms of "option value" (Baldwin and Clark, 2003),

[5]We refer to this concept from Fish (1980), cited by Brown and others in ther writing *Specialized Knowledge distributed in the classroom* (Compiled in Salomon 1993), in which "the participants are iniciated in discourse and academical activity rituals and, more specifically scientific ones".

that precisely it has to do with the possibility of choosing between different implementations of possible answers to the same (sub)problem. The more robust solution acted like an attractor in the previous described sense and it was chosen after being tested in the classroom, by means of the execution of its functionality in the environment for which it had been created and to be discuss by the students and the professor.

- The minimal size of the shared culture can be seen in the indicators of the competence and coherence of the agents system and also, these are interdependent of the modularity. As is well indicated by the multiagent systems literature (Durfee 1999), the coherence is easy to reach or is presupposed as a system condition. In this case a desing of a learning environment was made that embbed it, using a common project with individual subprojects, as well as evaluation instruments that valued and encourage the coherence. Once a big enough shared knowledge is reached and if the desing of the problem allows it, alternative solutions will rise, that can be interchanged in the environment. In the case of the students, it was easy to see the alternative solutions, in the case of the software agents, the coherence was *forced in the desing* (as well as in the classroom experience) and the competence and modularity can be seen in the amount of different messages (methods in OOP) that they were able to understand in common and in the way that these message were implemented.

- We had reached a limit regarding the modularity and this was because the desing of the *etoys* deliberately put a syntactic limit to the expressive possiblities of the language, in a way that extra complexities were not placed, for the children, for which they were not prepared, because they are the intended users of *etoys*. Nevertheless, this restricted the possibilities of interchanging small parts of the code that implement specific functionalities, so we need to share complete objects with all their methods or to use extended functionalities not provided by *etoys* that made complex the use of the interface at this point, and even with the help of the international *Squakers* community, which help us implementing some low level methods, going from tile based drag & drop programming to a syntactically simple code, but expressed in a way that we had never read previously was very difficult at this point[6] and so we can not modify, adapt or subdivide the code to guarantee the complete functionality of all the parts of the project.

Third moment: coherence, competence and modularity

In the third semester the experience was reinterpreted and redisigned to the light of the learnings of the two previous semesters. To not face the inconveniences of the syntactic limit of *etoys* it was thought in stablishing a bridge between them and the Smalltalk code that is underneath of their operation, a variant created in Squeak, was used, called Bots Inc (Ducasse, 2005), that is specially conceived for the teaching of programming concepts, at the same time that presents a graphical methaphor of the algorithms, from the idea of programming virtual software robots that can leave traces in an environment (in a similar inspiration to Logo, but from the Squeak's object multimedial environment). This time, it was about extending the robots' functionality in a way that were able to create a labyrinth and get out of it, being faced with different dangers (holes) and prizes (bonus). To create the shared culture (to increase the indixes of coherence and competence), it was not only replicated the educative environment in terms of the evaluation instruments, the methodologies, mediations and contents (eduwiki, Squeak, *Bookmorphs*, *etoys*, etc.), but it was also extended, to include new mediations, particulary the book and environment of Bots Inc, so we can access to more modular components of code and interchange them without major problems. Besides the attractors and bifurcations presented in the previous case, we can bold this moments of the classroom experience of the model:

[6]Esto se debía a que el código se había visto informálmente, pero que no había sido estudiado con detenimiento previamente al estar oculto, hasta el final del semestre, bajo la interface gráfica.

- The author moves from the minimal size of the shared culture, a group or environment indicator, inspired by the terminology of Axelrod, to coherence and competence, which are indicators associated to the individual agents. The shared culture of the system has a correlation with the coherence and competence of the agents.
- The modularity is specified as the characteristic wich previosly did not allow to interchange pieces of code, when the syntactical limit of the *etoys* was reached and is stablished a correlation between modularity and coherence: the agents will be more willing to assume active roles in the collective solution of the problem, if the pieces are well defined, and *small enough* (in some size metric) and they are interchangeables.
- A classroom space was set up with coherence (the instruments for evaluation, the mediations and dynamics valued and encouraged a lot the collective work, the individual apportations and the distribution), competence (similar contents were covered and evaluated until certain part of the semester) and the modularity (it was designed a collective problem with interdependent subproblems in an environment where the code was easily interchangeable).
- The first emergent phenomena started to show. The question for emergence in the classroom was one of the most difficult ones in the classroom experience. What was that thing that, in terms of the classroom experience of the model of collective solving of problems, was in the system, but was not in their individual components?. To appreciate the phenomena we adscribe the notion of classroom proposed by Brown and others (compiled by Salomon, 1999):

  > Theoretically we concibe the classroom as compossed by zones of proximal development (Vigotsky, 1978) by means of which the participants can move by different routes and at different speeds (Brown y Reeve, 1987). A zone of proximal development can include persons, adults and childs, with different degrees of specialiced knowledge, but can cover also artefacts such as books, videos, posters, scientific equipment and an computational context destined to support the intentional learning (Campione, Brown and Jay, 1992; Scardamalia and Bereister, 1991). A zone of proximal development is the activity region that the students can cross with the help coming from a supporting context that includes people but is not limited to them (Vigotsky, 1978). It defines the distances between the real levels of understanding and the ones that can be reached in collaboration with people or powerful artefacts. The zone of proximal development embodies a concept of disposition to learn that emphasizes superior levels of competence

  The emergent property is referred precisely to the solving of the problem, because this is a capacity of the system, that is no located individually in any of the agents. Of course, saying that, if there is competence and coherence, we all will solve a problem better that each of us individually, is a basic premise from wich is based the solving of problems in multiagent systems. The apportation of the model is the association of two characteristics of the multiagent systems, with one of the problem, which is the modularity. But the emphasis of the emergent phenomena, in terms of the classroom experience, is not there, but in the fact that, as a unity, the system becomes more competent (in a non linear way) with the increase of the individual competences and, in despite of the positive influence of this group's competence on individual competence, once the agents/students are considered alone, competence decreases considerably (as the evaluation in the course showed). The emergency arise because the development zones overlaped in the classrom modify the competence of the system for problems resolution and they leave "traces" in the individual competences. The grupal competence is an emergent phenomenon that arises from the fact that individual competences leave visible trails in the environment constituted by a modular problem.

The model

The multiagent system will be characterized by three elements for the solving of a problem:

- Coherence (*to want to solve it*): "Is the property or state of acting like a unity. The coherence deals with how good the system behaves like an unity. The evaluation criteria for coherence are, for instance, eficiency, quality of the solution and elegant degradation in presence of flaws" (Huhns, Michael N. and Stephens, Larry M., 1999). Observe that, even if the coherence is an emergent property of the system, it is required that each agent "wants" to act like a unity, according to the taks that was assigned to it. In other words, the group coherence emerges from the individual aportation of the agents.

- Competence (*to know to solve it*): "Is the hability of making well a task" (Durfee, Edmund, H., 1999). The coherence by itself is no enough to solve a problem, but also is required that each one of the agents be in conditionf of solving the part of the problem that was assigned to it. Again, is a global property that arises from the individual aportations of the agents.

- Modularity: (*decomposable in loosely coupled parts*): "Is refered as a general property of the complex systems, concerning the grade of *decomposability* of the system in loosely coupled parts made of strong coupled components. [...] A modular system, then, is represented as a complex of components or sub-systems where designers try to minimize and standardizing the interdependences between modules" (Schilling, 2000; cited by Narduzzo y Rossi).

Observe that, while the first two characteristics are refelated to the systems of agents, the last one is related with the problem, that in this case is modeled by the agents environment, and so, is also a characteristic related with the relation and interaction between the agents and the problem.

Is also worth to notice that the agents does not inhabitate a physical space, but a formal space, where the problem is modeled. The environment and the agents admit graphical bidimensional representations to visualize the model, but what is intended is to capture the generic characteristics of a problem in a formal space and also of the agents which are going to solve it.

The characteristics of competence and coherence, although owned by systems of agents, can be readed from premises associated to them as individuals that belong to a collective. That is, certain levels of individual coherence and competence, may give account of a set of shared values or *ethos* and common knowledges, which in turn may give account of complex phenomenon, like the size of the shared culture. So, the agents will have coherence and competence indexes that could represent that all have similar intentions or knowledges, or, that despite of having distinct knowledges, they are complementary in the problem's resolution. The fact that the agents communicate through the perceptions and changes in the environment, also represents characteristics that are present in the system rather than in the individual agents, that is associated to the interconnection and interdependence, which in turn can be associated with non explicit properties of the model, like the cultural ones.

We will preserve here the definition of an agent as a transformative function, that was presented previously, but we will sharp it to take in account of the characteristics of competence, coherence and modularity. We said that an agent was a function

$$Ag : \mathcal{R}^S \to Ac$$

from $\mathcal{R}^S$, the subset of all runings that end in states, to the set of actions Ac. This definition presented the domain and the range of the function, but did not define it in an explicit way (because it was a general definition). For the purposes of the model, we will proceed to define here the specific way in which the agent will "take a decision" about changing or not the environment using for that its levels of coherence and competence.

The algorithm of the agents families for every execution cicle, is showed in the proper figure.

```
Size of the problem > 0 ?
    Yes:
          Coherence level over a prefixed minimum ?
                Yes:
                   It's competent to aport of the solution of
                   that module of the problem?
                      Yes:
                         - Aport to the solution and change
                           the value of the state of the
                           environment where is located.
                         - Decrease the size of the problem
                           in one unity.
                      No:
                         - It moves to another "place" of
                           the enviroment state.
                No:
                   It moves to another "place" of the
                   enviroment state.
    No:
          it stops.
```

Figure 2: Algorithm executed in the model by the agents families

Computational implementation

For the implementation of the model Kedama was chosen (Yoshiki, no date), a multiagent simulation system, made on Squeak, that has been used to create models with reactive agents, wheter biological, as ants colonies (Yoshiki, no date) or social, as the Shelling segregation (Guillaume, 2006), among others. The reasons for the choice are not only associated with its fitness for the intended simulation, but also to the fact that it gives a visual programation system using tiles (*Etoys*), that it has an underlying object oriented language with simple and elegent syntax and also that squeak was the system in which the educative correlate and the classroom experience of the model, implemented in this section, was made

For a Kedama introduction, lets see what its author says: "Kedama is made up of several parts. At the bottom is the Kedama world itself, which appears as a large black box. On top of this is an invisible grid of squares. Each of the squares is called a patch, and the entire grid is controlled by patch variables that you can program. Patches are only found in Kedama and can be thought of as parts of a piece of graph paper with 100 x 100 cells (10,000 in total). Each cell can be assigned an integer value. Moving across these patches in the Kedama world are the turtle breeds, which are groups of turtles that act in the same way. You can decide how many different breeds there are, set how many turtles are in each breed, choose the color of the breed, and write the scripts that all turtles in each breed will follow. These turtles, moving across the Kedama world, can also read the value of each patch and respond to it. Finally, there are the individual turtles, or particles, that share the name of their breed. For example, the first breed that you create will be 'turtle1', so each turtle in this breed will follow instructions to turtle1. Turtles are given instructions as a breed, not as individual turtles" (Yoshiki, no date).

Two Kedama worlds, called `Kedama` and `Kedama1` where placed, representing the final state and the original state of the the agents environment. As it was said in the formal model, the difference between

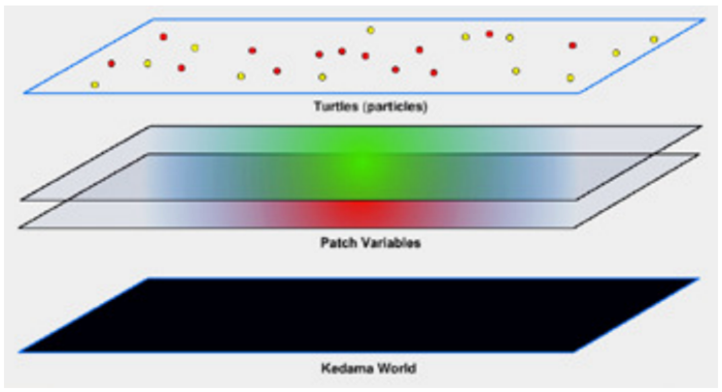
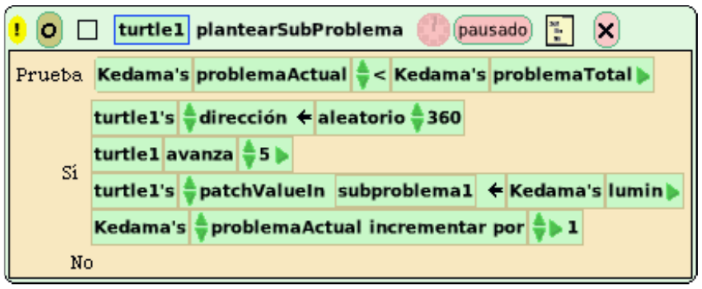


Figure 3: Layer architecture of the Kedama world.

this states is what constitutes the problem. In turn, each problem can be divided in subproblems, that in the Kedama world are represented by patchs. For purposes of this implementation a problem which consist of three subproblems or modules was placed. A simple metric for modularity was chosen saying that the modularity is equal to the number of modules.

Three tipes of variables where defined in the world which represents the objetive state: `problemaTotal`, `problemaActual` and `solucionActual`. The first one measures the size of the problem, namely the amount of values in the objetive state that are intented to be diffentet of the ones in the present state, the second measures the size of the problem when is raised, and the third one measures the size of the solution when the algorith is in execution.

Two kind of agents were programed (turtles in the Kedama vocabulary), ones that traveled the world to pose different problems and others that solve them. The algorithm which is the most important method of the agent which pose the problem is showed below:

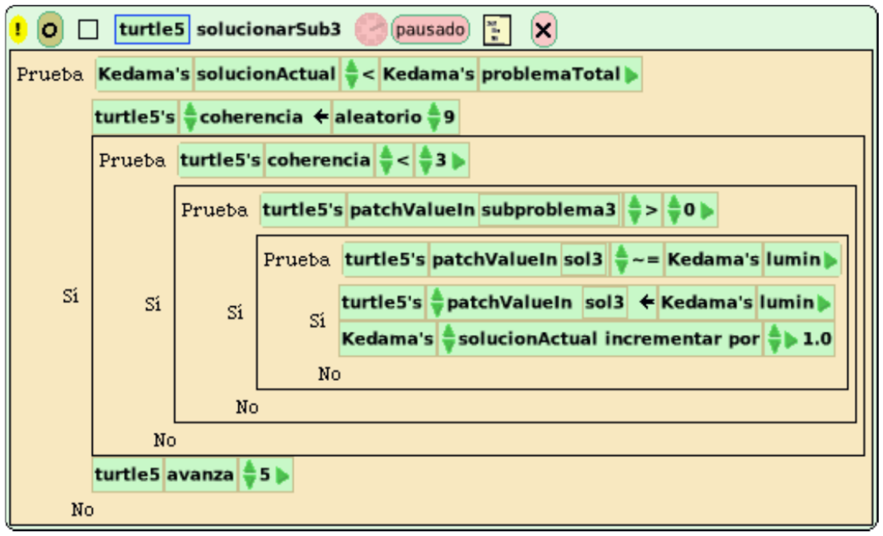


```
turtle1 plantearSubProblema   pausado
Prueba  Kedama's problemaActual < Kedama's problemaTotal
        turtle1's dirección ← aleatorio 360
        turtle1 avanza 5
Sí      turtle1's patchValueIn subproblema1 ← Kedama's lumin
        Kedama's problemaActual incrementar por 1
No
```

Figure 4: Method for posing the problem

Its behaviour is like follows: The first kind of agent compares if the problem that it has to help to propose is already of the desired size, and if this is not the case (i.e. `problemaActual` is smaller that `problemaTotal`) it choose a random place where to locate itself and writes in the `subproblema1` patch, which represents a module of the problem that is being posed, and changes the patch value, increasing `problemaActual` in one unit. In a similar fashion the other two kinds of agents propose the other subproblems. Note that here we have talked about kind of agents and not about specific agents, because we can can make that various agents of one kind only, help in posing of a subproblem, and in that case the different values between the initial state and the final state of the environment will be more sparse in all the environment.

The solution of the problem is the one implemeted by Kedama in the algorithm of following section and is the core of this writing. Two kind of agents for problem solving have been placed to solve three kind of subproblems. The green ones, which only can solve type 3 subproblems (i.e. problems placed is the space of the patch 3) and the others, the red ones, which can solve type 1 or type 2 subproblems (i.e. placed in the spaces of the patch 1 and 2). In the vocabulary of the model the red agents are more competent that the green agents. Following the hypothesis of Narduzo and Rossi (2003) that says that more modular the problems affect positively the "willing" of the agents in a collective for solving them (because they can take care of smaller sections), a random factor, which models coherence and is related with modularity in this way: if the random number, is smaller than the modularity, then the agent will try to verify its competence (if it can apply its knowledge to the solutions) and in the case of being competent and the value in the solution space is without solution, it will contribute to the solution changing the state in the patch which models the subsolution and it will indicate this by increasing the variable `solucionActual` in one unit. The algorithm, implemented in Kedama, is as follows:

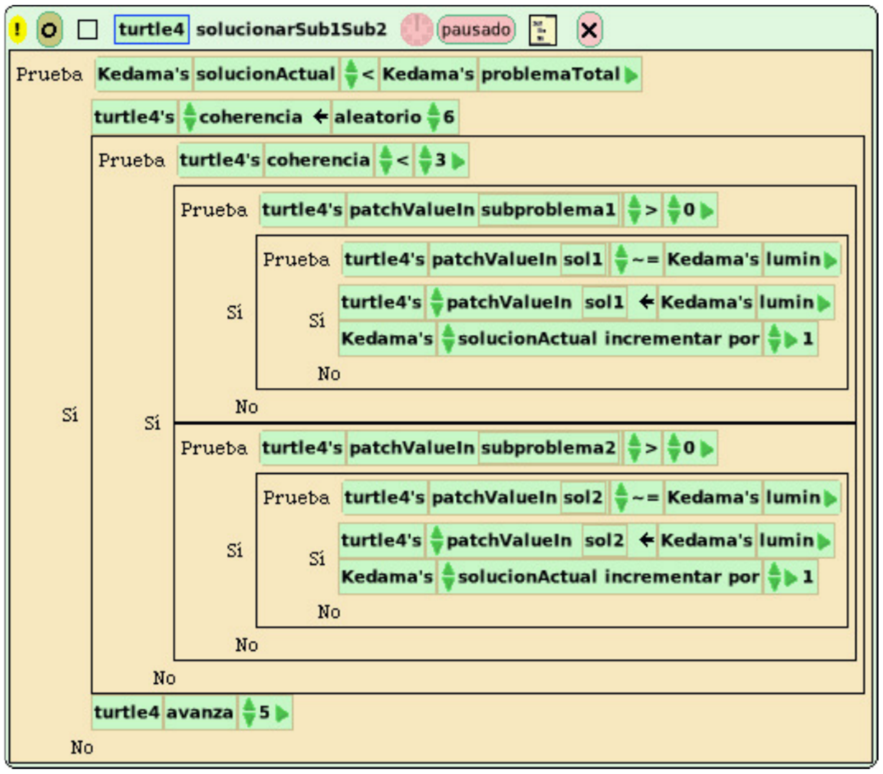


```
turtle5 solucionarSub3   pausado
Prueba Kedama's solucionActual < Kedama's problemaTotal
Sí  turtle5's coherencia ← aleatorio 9
    Prueba turtle5's coherencia < 3
    Sí  Prueba turtle5's patchValueIn subproblema3 > 0
        Sí  Prueba turtle5's patchValueIn sol3 ~= Kedama's lumin
            Sí  turtle5's patchValueIn sol3 ← Kedama's lumin
                Kedama's solucionActual incrementar por 1.0
            No
        No
    No
    turtle5 avanza 5
No
```

Figure 5: Algorithm of the problem's solution of the model presented in this work implemented in Kedama. Nested test must be used to simulate the logical operator "and", because the *Etoys* graphical programming system in its actual version doesn't support it in graphical fashion.

The algorithm of the red agents is similar in essence, with the differente that, being mode compentent, they must make moder nested test, which are associated to where thet can apply their knowledge (if in the subproblem 1 and/or in the subproblem 2). The implementation in Kedama is showed as following:

Finally an interface was implemented in the Squeak environment with a set of controls which let run varios simulations easily, specifing the size of the `problemaTotal` parameter, and controlling the times when the methods of the agents families for posing problems and running solutions were invoked. A screenshot can be seen in the figure 6 and 7.



```
turtle4 solucionarSub1Sub2 pausado
Prueba Kedama's solucionActual < Kedama's problemaTotal
Sí
    turtle4's coherencia ← aleatorio 6
    Prueba turtle4's coherencia < 3
    Sí
        Prueba turtle4's patchValueIn subproblema1 > 0
        Sí
            Prueba turtle4's patchValueIn sol1 ~= Kedama's lumin
            Sí
                turtle4's patchValueIn sol1 ← Kedama's lumin
                Kedama's solucionActual incrementar por 1
            No
        No
        Prueba turtle4's patchValueIn subproblema2 > 0
        Sí
            Prueba turtle4's patchValueIn sol2 ~= Kedama's lumin
            Sí
                turtle4's patchValueIn sol2 ← Kedama's lumin
                Kedama's solucionActual incrementar por 1
            No
        No
    No
    turtle4 avanza 5
No
```

Figure 6: Implementation of the algorithm of the model, for more competent agents

Conclusions and recomendations

The collective solution of problems indeed can be characterized, in a general way, by three properties: coherence, competence and modularity. The first two are properties of the systems of agents, while the last one in a property of the problem (environment). This is, as far as the literary revision showed, the first model that has these three properties together. Coherence and competence can be seen in terms of a property of the individual agents in the system, that are related with the minimal size of the shared culture, considering culture from the perspective of Axelrod (2004), i.e, as a set of values that have a set of values. When variables have the same values, we talk about regions of culture and this imply, in terms of coherence and competence, shared intentions and vocabularies and this is explicit in more complex constructs (methods and messages in the object oriented programming, forums, wikis, and so on, in the communities of practice).

Microchanges generate macrobehaviors. In this sense, it validades what was said by Schelling is his segregation model, and is applied also to the collective resolution of problems. This is made explicit by:

- how the classroom experience tried to propagate shared *ethos* and *knowledge*, in the individual students, using: curricular contents, tecnological computer instruments auto and hetero evaluation mechanism and
- how the individual publication and use of the mediations was considered, even though we wanted to evaluate a collective process finally, without straying the attention, of the individual process and contributions to the subproblems.

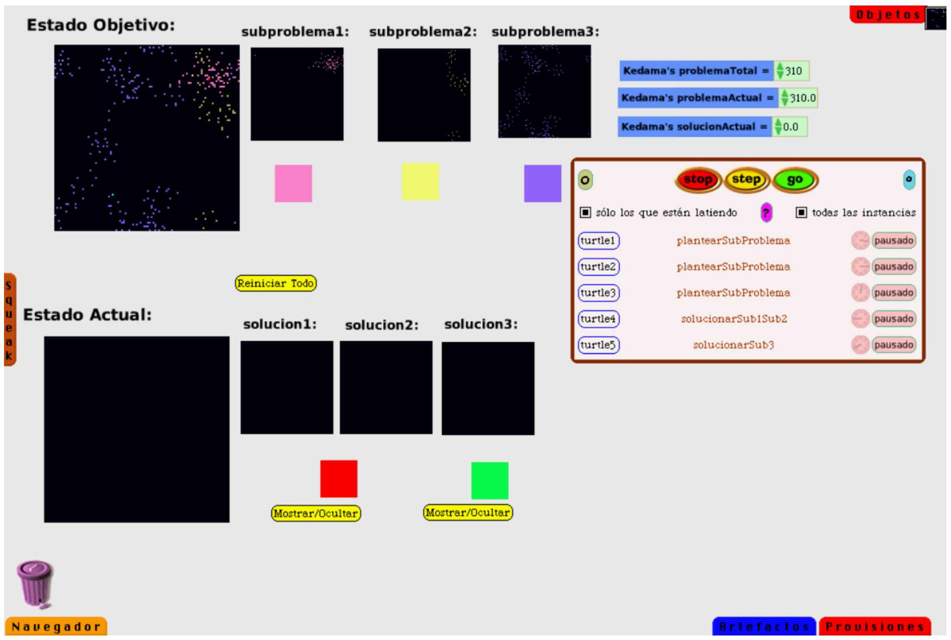
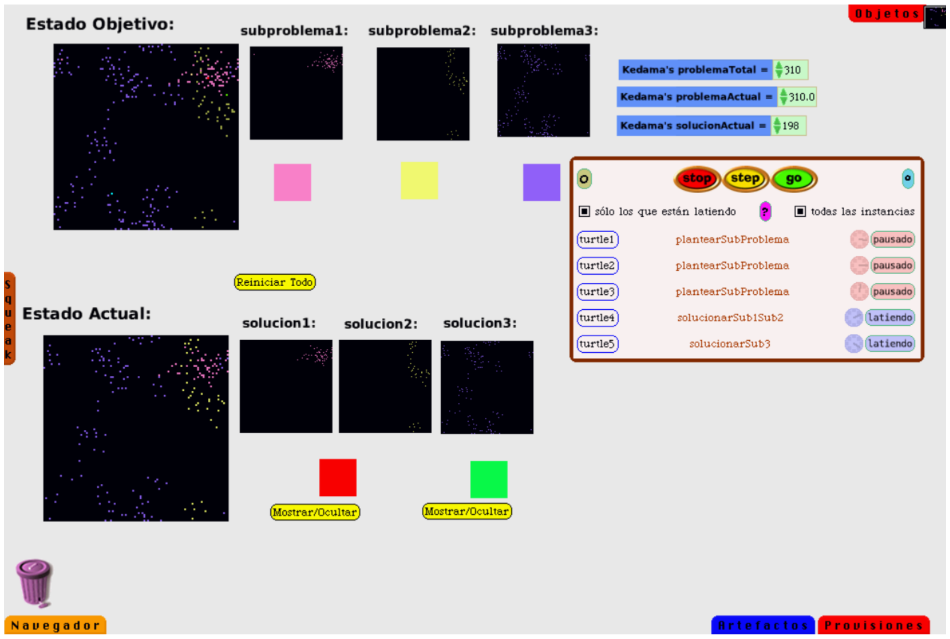

Estado Objetivo:
subproblema1:
subproblema2:
subproblema3:
Objetos
Kedama's problemaTotal = 310
Kedama's problemaActual = 310.0
Kedama's solucionActual = 0.0
stop
step
go
sólo los que están latiendo
todas las instancias
turtle1 plantearSubProblema pausado
turtle2 plantearSubProblema pausado
turtle3 plantearSubProblema pausado
turtle4 solucionarSub1Sub2 pausado
turtle5 solucionarSub3 pausado
Reiniciar Todo
Squeak
Estado Actual:
solucion1:
solucion2:
solucion3:
Mostrar/Ocultar
Mostrar/Ocultar
Navegador
Artefactos
Provisiones

Figure 7: The execution environment for the simulations of the model. It can be seen the objetive state of the environment, which, in turn, is made of three subproblems or modules and, bellow each one, exemplars of the families of agents which help to pose them. At the right the values of the more important variables of the environment are located and a tool for invoking the individual execution for posing problems or their solutions. In the bottom the actual state of the environment can be seen, before running the solution of the problem, along with the subsolutions and copies representing of the families of agents that solve them.

The emergent phenomenon of the collective resolution of problems, that was beyond the individual capacities of every agent was one of the evidences of the macrobehavior. The microchanges happen because every one of the agents incorporate the intentions and vocabularies as permanent rules in their interaction with the environment (as in the case of the ant colony simulation) or because the agents between them, in their interaction, the rules (as in the epidemy simulation). It would be interenting to dig more in the educative correlate of this conclusion from the perspective of the ideas of proximal developmen zones and their scaffolding in the student-student and student-environment relationship and the metaphor used of the classroom as a space where multiple proximal delevoment zones are overlaped.

The closer they are from finding the *ideal solution* the longer the agents take to make contributions to that solution. This was one of the most interesting emergent phenomenon in the runnings of computer simulation of the model in several times. Because the agents have already contributed their knowledges in their walks on the solution space, now its about going to that specific place of the space where the contribution is not yet done, however, most of the exploration happen in places already traveled in the solution space, that is a bigger place that the one where the specific contribution is still lacking. As was seen in the classroom correlate, once we had integrated the most part of the prototype of the collective project, the specific errors depuration and the resolution of unexpected codependencies took much more time that the original integration and in fact was not all finished, because of the cost of getting a minimal size of the shared culture. So, its important, once the coherence and competence of the system have been reached, to create proper curricula to let the system to continue its existence



Kedama's problemaTotal = 310
Kedama's problemaActual = 310.0
Kedama's solucionActual = 198

sólo los que están latiendo · todas las instancias

turtle1	plantearSubProblema	pausado
turtle2	plantearSubProblema	pausado
turtle3	plantearSubProblema	pausado
turtle4	solucionarSub1Sub2	latiendo
turtle5	solucionarSub3	latiendo

Figure 8: Simulation running the algorithms for collective problem solving in the model, what can be seen because the little clocks that control the read and green families of agents have the label ¨latiendo ¨ (running) in the tool for script control in the right and are colored in blue. See how the solutions 1 to 3 in the bottom are starting to look like the subproblems of the objetive state in the upper part.

and functioning to make more specific contributions to more detailed (sub)problems.

Narduzzo and Rossi (2003) show how modularity is an architectonic characteristic that, in the specific case of software development, improves the chances of the making of contributions. In the more general terms of the model presented here, we could say that modularity possitively affects the coherence of the agents. This was cleare saw in the classrom correlate, when, once we have reached the syntactic limit provided by *Etoys*, was very difficult to make the integration of the individual subproblems which made part of the general problem, while using *Bots Inc* the possibility of deconstruction and integratin was increased and in the end we could solve the proposed collective problem.

The progressive increase of competence is favored by continuous and deconstrutible metaphors. The basic concepts of object oriented programming (inheritance, encapsulation, polymorphism and agregation) were addressed several times using different but *continuous* and complementary mediations and activities: at the begining the construction of an interactive *bookmorph* was offered (to stablish continuity with the usual previous activities, like the creation of presentations in *PowerPoint* u *OpenImpress*), then the same conceptual basic elements were used in the construction of an interactive game to emphasize the programming elements and to find the limits of the *Etoys* interface metaphor and later we passed to *Bots Inc* where the graphical correlate of the algorithmic and object oriented programming concepts were sustained, but we have access to simpler, lower leven and powerful sintactic constructs. The deconstrutible character of the metaphor is related with the increases of competence of the cognitive agent, and thanks to that increases, the modularity is also increased and so the option value in the solutions of the problem.

The unforeseen codependencies are part of a decomposition problem (Narduzzo y Rossi, 2003) and is good to have integrating functions that help to solve them (for example the bigger method that invoked the particular methods in the classroom experience). Some of the functionalities of the methods-modules, that solved specific problems, once integrated in the solution of the general problem showed unforeseen codependencies between the subproblems that affected the integration and, in this case, what we did was to relegate the conflictive behaviors to specific methods and integrate the part that solve the codependencies to the general method. To resume, the unforeseen codependencies in the solutions to the subproblems can be solved putting the conflicting functionality in the subproblems and integrating it in a communication mechanism between them.

The external mechanism of representation and the competence of the system are interdependent. The bigger the competence of the system, the more complex, dense, and frecuently used, the external representation mechanism become. In the case of Eduwiki it can be seen how we pass from personal pages without comments to pages with superficial comments and then to the constitution of a virtual forum, where common advances were shared, general answers to the integration problems and the groups help each other in the integration, and in the same way external representation systems as *Internet Archive* began to be used, which let share all the state of the *Bots Inc* system, when integration requiered this.

Some activities that in the school system has been encouraged usually from the perspective of the individual work may be reinterpretated and be befinefied under the light of distributed cognition, for example, the problem solving in general and computer programming in particular (that is seen as an activity which is made and learn by a programmer in "solo" in front of his/her machine). In this sense, this work is subscribed to what is said by Dillebourg (1996), about that the primary functions of the learning envionment (including diagnosis, tutoring and explanation) are primary distributed and contribute with models and mediations that explicitate how this happen with the problem solving in particular. We also share the posture that this has consecuences in the design of educative systems and, in fact, the classroom correlate, showed a possible design that may be used in order to make clear and encourage the collective problem solution.

Classroom and evaluation desings may encourage the coherence and compentece increasing the awareness of the system about itself (strong emergence). The evaluation mechanish used in the classroom experiences also evolved and had a diagnostic character encouraging individual indicators of coherence and competence (dynamic evaluation favours coherence and competence in the agents). The first because if and already evaluated knowledge get better, the calification in the instrument reflected that, and the second because a constant element of the evaluation was the comments made by the students in the pages of each others and the integration of the individual work in the common project, so in this way the reading between them and the contrubition of collective solutions was encouraged.

Is possible to create curricular experiences that are related with teaching common knowledge and after that to take different subproblems affecting the expertise of the takers or users. In that sense curriculum could be rethought in such wat that *courses* offer both, similar curricular content (which configure minimal size of the shared culture), as also levels of specilization inside them, so is possible to generate variations on that common knowledges, applied to specific subproblem or that let to use other knowledges in the subproblems. This helps with the configuration of a community of discourse and practice, because generates an equilibrium between the common and complementary knowledges.

Posterior studies about the same theme may improve the computational model and the applications and dialogs between the classroom experience, particulary regarding the desing of learners networks that solve problems collectively and of the mediation that encourage them, emphasizing two aspects: the pass from the individual agento to the collective agency and the validation, not only functional, but also comportamental, of the model or its modifications, for the purpose of being closer to such validations and make more explicit the scopes and limitations of the model.

Bibliography

- A Guide for Writing Research Papers based on Styles Recommended by The American Psychological Association

- Axelrod, Robert (2004). La complejidad de la cooperación. Fondo de Cultura Económico. Argentina.

- Baldwin, Carliss Y. y Clark, Kim B. (2003). The Architecture of Cooperation: How Code Architecture Mitigates Free Riding in the Open Source Development Model. Disponible en Internet:

  `http://opensource.mit.edu/papers/baldwinclark.pdf`

- Bellifemine, F.; Caire, G.; Poggi, A. y Rimassa, G. (sin año). JADE A White Paper. Disponible en Internet: `http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf`

- Bordini, R y Hübner, J. (2005). Jason A Java-based agentSpeak interpreter used with saci for multi-agent distribution over the net. Disponible en Internet:

  `http://jason.sourceforget.net/jason.pdf`

- Brow, Matthew. Cartesian Cognitivism and Its Discontents. (2004). Disponible en Internet. `http://thm.askee.net/articles/cartesian-cogn.pdf`

- Cavallo, David y el Future of Learning group at the MIT Media Lab (2004). Models of growth – towards fundamental change in learning environments. En BT Technology Journal. Vol 22 No 4 . October 2004.

- Cederman, Lars- Erik y Gulyas, Laszlo. (2001). Tutorial Sequence for RePast: An Iterated Prisoner's Dilemma Game. Disponible en Internet:

  `http://www.econ.iastate.edu/tesfatsi/RepastTutorial.IPD.pdf`

- Collier, Nick (sin fecha). RePast: An Extensible Framework for Agent Simulation. Social Science Research Computing. University of Chicago Chicago, IL. Disponible en Internet:

  `http://www.econ.iastate.edu/tesfatsi/RepastTutorial.Collier.pdf`

- Cost Scott y otros (sin fecha). TKQML: A Scripting Tool for Building Agents. Disponible en Internet: `http://citeseer.ist.psu.edu/cost97tkqml.html`

- Cunningham, Ward; Jeffries, Ron; Kessler, Robert R. Williams, Laurie (2000) Strengthening the Case for Pair Programming. En IEEE Software July/August 2000.

- Cycorp (2002). OpenCyc Tutorial. Disponible en Internet:

  `http://opencyc.org/doc/tut/?expand_all=1`

- Dastani, Mehdi y otros. (sin año). Programming Agent Deliberation. An Approach Illustrated Using the 3APL Language. Disponible en Internet:

  `www.cs.uu.nl/3apl/publication/agentdeliberation.pdf`

- Dastani, Mehdi (2004). 3APL Platform User Guide. Disponible en Internet:

  `http://www.cs.uu.nl/3apl/download/java/userguide.pdf`

- Dillenbourg, Pierre. (1995). From Mutual Diagnosis to Collaboration Engines: Some technical implications of distributed cognition on the desing on interactive learning environments. Edited transcrip of an invited talk at the World Conference on Artificial Intelligence in Education. Disponible en Internet.

  `http://tecfa.unige.ch/tecfa/publicat/dil-papers-2/Dil.7.2.9.pdf`

- Ducasse, Stéphane (2005). Squeak: Learn Programming with Robots. Apress California.

- Evans, Phillip y Wolf, Bob (2005). Collaboration Rules. En: Hardvard Business Review.

- Freeh, Vincent; Madey, Gregroy y Tynan, Renee. (Sin Fecha). Modeling the Free/Open Source Software Community: A Quantitative Investigation. Disponible en Internet:

  `http://www.nd.edu/~oss/Papers/BookChapter.pdf`

- Galoppini, Roberto y Garzarelli Giampaolo (2003). Capability Coordination in Modular Organization: Voluntary FS/OSS Production and the Case of Debian GNU/Linux. Disponible en Internet:

  `http://econpapers.repec.org/paper/wpawuwpio/0312005.htm`

- Guillaume, Blondel y Ludwig, David (2006). Modèle de ségrégation de Schelling Rapport de projet Intelligence Artificielle distribuée. Universidad de Caen.

- Harasim, Linda y otros. (1998). Redes de Aprendizaje. Gedisa. España.

- Heylen y otros. (2004). Research Proposal for a "Geconcerteerde Onderzoeksactie" Modelling the Emergence and Evolution of Distributed Cognition. Disponible en Internet:

  `http://pespmc1.vub.ac.be/Papers/GOA-project.pdf`

- Hutchins, E. (1995). Cognition in the Wild. M.I.T. Press. Cambridge, Massachusetts.

- Josephson, Susan (2000). Thinking like a Machine. Knowledge Systems approach to Artificial Intelligence. Disponible en Internet: `http://www.cse.ohio-state.edu/lair/Main/TLM.html`

- Kerckhove, Derrick. (1999). Inteligencias en conexión. Gedisa. España.

- Kumar, Ashwani. (2005). i-Seek: An Intelligent System for Eliciting and Explaining Knowledge. Disponible en Internet: `http://agents.media.mit.edu/projects/iseek/ashwani-ms.pdf`

- Kumar, Ashwani; Sundararajan, Sharad C. y Lieberman Henry. (sin fecha) Common Sense Investing: Bridging the Gap Between Expert and Novice. Disponible en Internet:

  `http://agents.media.mit.edu/projects/investing/lb366-kumar.pdf`

- Laird, John; Lehman, Jill Fain y Rosenbloom, Paul (2006) A gentle introduction to soar, An architecture for human cognition: 2006 Update. Disponible en Internet:

  `http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/GentleIntroduction-2006.pdf`

- LinGO Matrix Project (2004). Grammar Engineering, Introduction, overview, HPSG basics. Disponible en Internet: `http://courses.washington.edu/ling471/0329.pdf`

- Liu, Hugo y Lieberman, Henry (sin fecha) Metafor: Visualizing Stories as Code. Disponible en Internet: `http://web.media.mit.edu/~hugo/publications/papers/IUI2005-metafor.pdf`

- Liu, Hugo; Lieberman, Henry y Selker, Ted (2003). Visualizing the Affective Structure of a Text Document. Proceedings of the Conference on Human Factors in Computing Systems, CHI 2003, April 5-10, 2003, Ft. Lauderdale, FL, USA. ACM 2003, ISBN 1-58113-637-4, pp. 740-741.

- Liu, Hugo; Lieberman, Henry y Selker, Ted. (2002). GOOSE: A Goal-Oriented Search Engine With Common- sense. In De Bra, Brusilovsky, Conejo (Eds.): Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, Malaga, Spain, May 29-31, 2002, Proceedings. Lecture Notes in Computer Science 2347 Springer 2002, ISBN 3-540-43737-1, pp. 253-263.

- Liu, Hugo y Sing, Push (2004) ConceptNet – a practical commonsense reasoning tool-kit. En: BT Technology Journal . Vol 22 No 4 . Octubre 2004.

- Liu, Hugo y Sing, Push. (sin fecha). Commonsense Reasoning in and over Natural Language. Disponible en Internet: `http://web.media.mit.edu/~push/CommonsenseInOverNL.pdf`

- Macal, Charlas M. y North, Michael J. (2005) Tutorial on agent-based modelling and simulation. Center for Complex Adaptive Agent Systems Simulation (CAS2) Decision & Information Sciences Division Argonne National Laboratory Argonne, IL 60439, U.S.A.

- Maxwell, John W. (2006). Tracing the Dynabook: A study of Technocultural Transformations. University of British Columbia. Disponible en Internet:

  `http://thinkubator.ccsp.sfu.ca/Dynabook/dissertation`

- Minsky, Marvin (1986). Society of Mind. Simon & Schuster. Sidney, Australia.

- Minsky, Marvin y Sing, Push (2003). An Architecture for Combining Ways to Think. En Proceedings of the International Conference on Knowledge Intensive Multi-Agent Systems. Cambridge, MA. 2003.

- Musa, Rami; Kulas, Andrea; Anguilete, Yoan y Scheidegger, Madleina (sin fecha). GloBuddy, a Tool for Travelers. Disponible en Internet:

  `http://www.media.mit.edu/~lieber/Teaching/Common-Sense-Course/Projects/GloBuddy/GloBuddy.pdf`

- Narduzzo, Alessandro y Rossi, Alessandro (2003). Modularity in Action: GNU/Linux and Free/Open Source Software Development Model Unleashed. Universidad de Bologna y Universidad de Trento. Disponible en Internet:

  `http://rock.cs.unitn.it/file/NarduzzoRossiFOSS2003.pdf`

- Pan, Denis (2004). L'émergen

- Pfeifer, Rolf y Scheier Christian. (1999). Understanding Intelligence. M.I.T. Press. Cambridge, Massachusetts.

- Salomon, Gavriel (compilador) (1993). Cogniciones distribuidas. Consideraciones psicológicas y educativas. Amorrortu Editores. Buenos Aires.

- Sicilia, Miguel Angel (sin fecha). Sobre la concepción de conocimiento en el proyecto OpenCyc. Disponible en Internet:

  `http://serbal.pntic.mec.es/~cmunoz11/sicilia36.pdf`

- Sing, Push (2003). Examining the Society of Mind. Disponible en Internet:

  `http://web.media.mit.edu/~push/ExaminingSOM.html`

- Leigh Tesfatsion. (sin fecha). Agent-Oriented Programming. Department of Economics, Iowa State University. Disponible en Internet: `http://www.econ.iastate.edu/tesfatsi/AOPRepast.pdf`
- Weiss, Gerhard (1999). Multiagent Systems A modern Approach to Distributed Artificial Intelligence. M.I.T. Press. Cambridge, Massachusetts.

Williams, Laurie (2000). Strengthening the Case for Pair Programming. IEEE Software Magazine. EEUU. Disponible en Internet:
`http://www.computer.org/software/so2000/pdf/s4019.pdf`

- Woolridge, Michael, J. (1996) An introduction to multiagent systems.
- Yoshiki, Ohshima (sin fecha). Kedama: A massively-parallel tile-scriptable particle system. Disponible en Internet: `http://www.is.titech.ac.jp/~ohshima/squeak/kedama/`
- Yoshiki, Ohshima (sin fecha). Fun with Kedama. Disponible en Internet:

  `http://www.squeakland.org/fun_projects/kedama/kedma_getstart.htm`