Resolución Colectiva de Problemas desde Modelos Multiagente: un diálogo entre la teoría y el aula

por Offray Vladimir Luna Cárdenas

Correo-e: offray.luna@javeriana.edu.co
Web: www.el-directorio.org/Offray

Resumen

El presente artículo tiene como objeto caracterizar la resolución colectiva de problemas desde modelos multiagente. Desde el abordaje teórico se asumió una postura considerando la escuela socio histórica rusa, la cognición distribuída y la teoría de sistemas multiagentes y, en cuanto a lo metodológico, se optó por el modelaje y la simulación como una forma de aproximarse a la compresión y a la caracterización, al mismo tiempo que se establecía un correlato de aula que dialogara con el modelo, para que mutamente se inspiraran y cambiaran. Se llegó a un modelo que caracteriza la resolución colectiva de problemas en sistemas multiagente a través de dos parámetros de los sistemas de agentes: coherencia y compentencia, y uno del problema: modularidad, y se valido el modelo al evidenciar fenómenos emergentes tanto en el modelo como en su correlato de aula.

Palabras clave: cognición distribuida, resolución de problemas, sistemas multiagente, simulación, escuela sociohistórica, Kedama, Squeak, Bots Inc, diseño de entorno, educación.

Abstract

The present article has as objetive 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 both on the model and in the classroom experience.

Keywords: distributed cognition, problem solving, multiagent systems, simulation, sociohistoric school, Kedama, Squeak, Bots Inc, ambient design, education.

Introducción

¿Como se solucionan problemas en colectivo?. Esta es la pregunta que este artículo busca abordar. La pertinencia de la misma está subrayada por el hecho de que la escuela socio histórica rusa y la de la cognición distribuída han puesto de manifiesto la importancia del entorno, las mediaciones, los otros y lo sociocultural en el desarrollo cognitivo y también el hecho de que la inteligencia no sólo están "dentro" de la cabeza, sino que ella emerge y se da en y gracias a lo colectivo. El auge del software social, en sitios como la Wikipedia, SourceForge y sus proyectos de software libre, Amazon, Youtube, muestran explicitaciones informáticas de estas inteligencias colectivas. Cómo se da y opera esta inteligencia, cómo se relacióna con lo individual, cómo se establecen "continuos" y otras tantas inquietudes son parte de este campo relativamente novel, con importantes desafíos teóricos y abrumadoras consecuencias prácticas, en particular y desde el tema que nos atañe, en lo cognitivo y lo educativo. Este artículo da cuenta de una aproximación al estudio de esta inteligencia colectiva y sus continuos, específicamente indagando por la resolución colectiva de problemas.

Para dicha indagación se han tomado dos caminos dialogantes: un modelo algorítmico con implementación computacional y un correlato de aula. El último se ha abordado con un enfoque que se fundamente en la escuela postvigostkiana y de la cognición distribuída y en experiencias de trabajo con estudiantes universitarios durante tres semestres, usando mediaciones tecnológicas de software social (Wikis) y entornos de programación orientado a objetos, multimediales y de programación gráfica (Squeak - eToys) o con un fuerte elemento visual (Bots Inc). El primero ha asumido metodológicamente la aproximación desde la modelación sintética basada en multiagentes, del que algunos autores (Pfeifer y Scheier, 1999 y Macal y North, 2005) mencionan ser un tercer enfoque entre lo analítico y lo sintético o entre lo inductivo y lo deductivo. Este lugar se hibrida aún más si se consideran las interacciones entre el modelo formal y la experiencia de aula y fue el escenario fascinante donde se dió esta indagación, entre (des)encuentros, reconcialiciones y momentos ajá, fruto de la inspiración, pero sobre todo y como dice el refrán, de la transpiración.

Se presentan primero y rápidamente los elementos teóricos que permiten formular el modelo. En este punto, la presentación formal se detiene y se aborda el diálogo entre el modelo formal y la experiencia educativo que avanzaron en paralelo y de modo dialogante. Se consideran rápidamente elementos de la experiencia educativa para dar finalmente paso el modelo completo y por último se presentan las conclusiones y recomendaciones.

Elementos teóricos

Un agente se considera como una entidad que está en un entorno y está en condiciones de realizar acciones sobre él a fin de lograr un objetivo o posible estado final del entorno. En cuanto a los entornos, estos se consideran

- Accesibles/Inaccesibles : Si para un agente toda la información del entorno está disponible, se dirá accesible, de lo contrario se considerará inaccesible.
- Determinísticos/No determinísticos : Si las mismas acciones de los agentes en el entorno producen los mismos resultados independientemente de otros factores, se llamarán determinísticos y en cualquier otro caso serán no determiníticos.
- Estáticos/Dinámicos: Si el entorno sólo cambia por la acción de un agente, se dirá estático. Si los cambios ocurren además por factores que se encuentran fuera de la influencia de un agente, entonces se dice que el entorno es dinámico.
- Discretos/Continuos: Un entorno discreto es aquel en el cual las acciones y percepciones sobre él son un número finito.

Existen varias taxonomías también asociadas a los agentes. Se dice que los agentes pueden ser: biológicos; robóticos (como los que se dan en la investigación y las aplicaciones industriales); Computacionales y Biológico Culturales. Wooldridge (1999) señana tres características importantes en los agentes a los que vayamos a adscribir comportamiento inteligente:

- Reactividad: gracias a ella un agente responde en una manera oportuna a los cambios del entorno, a fin de lograr su objetivo.
- Proactividad: se refiere al hecho de que los agentes pueden tomar al inciativa a fin de establecer sus objetivos de diseño y mostrar un comportamiento guiado por metas.
- Habilidad Social: tiene que ver con la posibilidad de los agentes de interactuar con otros agentes.

Para llegar a la definición formal de agente, de modo constructivo, se presentarán las definiciones formales de los elementos que se involucran en ella.

Definición. ***Estados***: Un entorno puede estar en un conjunto discreto E de estados instantáneos que se simboliza

$$E = \{e_0, e_1, ...\}$$

obsérvese que se ha escogido un entorno como un conjunto discreto, sin embargo esto no afecta la posibilidad de que nuestros agentes operen en entornos continuos, sin pérdida de generalidad, pues supondremos que las percepciones de los agentes sobre entornos continuos, pueden ser mapeados, para efectos de su acción, en estados discretos.

Definición. ***Acciones***: Las acciones que el agente puede hacer las definimos como el conjunto finito:

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

Hemos dejado por fuera del modelo, hasta el momento, el caracter autónomo del agente y su objetivo, pero nos ocuparemos de ellos más adelante, para concentranos en el entorno y las acciones, por lo pronto.

Definición. ***Ejecución***: Diremos que las acciones de los agentes modifican el entorno a traves de las ejecuciones, que son secuencias intercaladas de estados y acciones, dado que cada acción de un agente sobre el estado de un entorno, produce un nuevo estado. Se tiene entonces que una ejecución r es:

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

y, continuando con la propuesta de Wooldridge, definimos:

- $\mathcal{R}$: el conjunto de todas las posibles ejecuciones r
- $\mathcal{R}^{Ac}$: el subconjunto de $\mathcal{R}$ cuyas ejecuciones terminan en acciones.
- $\mathcal{R}^{E}$: el subconjunto de $\mathcal{R}$ cuyas ejecuciones terminan en estados.

Definición. ***Función transformadora de estados***: Una función transformadora de estados τ, que representa las acciones de un agente en un entorno, puede ser vista como:

$$\tau: \mathcal{R}^{Ac} \rightarrow \mathcal{P}(E)$$

que envia un subconjunto de ejecuciones, que terminan en acciones, en un subconjunto de estados, resultado de dichas acciones sobre los estados previos.

Definición. ***Entorno***: Se define un entorno Ent como una tripleta de la forma $Ent = \langle E, e_0, \tau \rangle$, donde E es un conjunto de estados del entorno, $e_0 \in E$ es el estado inicial y τ es una función transformadora de estados. Nótese como esta definición enfatiza los posibles estados del Entorno, el estado inicial y la función que transforma de este estado inicial a cualquiera de los estados posibles.

Definición. ***Agente***: Los agentes que habitan el entorno se definen como funciones que van de las ejecuciones que terminan en estados a las acciones. Es decir la función principal de un agente es transformar estados del entorno en acciones. Las acciones resultantes a su vez generan nuevos estados que influencian de modo posterior al agente (esta es la noción de interacción agente - entorno). Formalmente se tiene:

$$Ag: \mathcal{R}^{E} \rightarrow Ac$$

El hecho de que la función, que define al agente, tenga en cuenta las ejecuciones que terminan en estados y no sólo los estados, quiere decir que los agentes tienen algún sentido de la historia (salvo en el caso de los agentes púramente reactivos, que mencionaremos brévemente más adelante).

Ahora bien, los estados del entorno no son equivalentes a lo que el agente «sabe» de éste, pues, en general, los agentes habitan entornos inaccesibles. Esto introduce la noción de percepción. Una percepción es aquello que el agente capta de los estados del entorno (que no forzósamente tiene que ver con los estados reales) a través de una función sensora

$$\text{sensor}: E \rightarrow \text{Per}$$

y la acción es un mapeo que van desde las percepciones a las conjunto de acciones:

$$\text{accion}: \text{Per} \rightarrow Ac$$

un agente, entonces, es un par sensor-acción, $Ag = \langle \text{sensor}, \text{accion} \rangle$.

Un problema se definirá en el modelo como una diferencia entre el estado objetivo de un entorno multiagentes y el estado actual del mismo y la resolución del problema será la ejecución de unas acciones cuya finalidad sea hacer coincidir el (futuro) estado objetivo del entorno y el estado percibido como actual[1].

Definición. ***Problema***. En términos de las notaciones antes establecidas, diremos, entonces, que un problema P es la diferencia entre un estado e_0 y un estado e_f del entorno Ent. A su vez el estado e_0 está caracterizado por un conjunto ordenado de valores $e_0 = \{v_1, v_2, ..., v_n\}$ y el estado e_f esta caracterizado por un conjunto ordenado de valores $e_f = \{v'_1, v'_2, ..., v'_n\}$. Se dice que el problema persiste si existe al menos un $v_i \neq v'_i$ para i dentro de los índices de los estados e_0 y e_f; y se dice que el problema fue solucionado si $v_i = v'_i$ para todo i en los índices de los estados e_0 y e_f. El tamaño del problema es la cantidad de valores para los cuales $v_i \neq v'_i$. Es decir, un problema solucionado es aquel que tiene tamaño 0.

Diálogo entre el modelo y la experiencia educativa

El correlato educativo del modelo y el modelo mismo se desarollaron en paralelo durante un tiempo de tres semestres (dos del 2006 y el primero del 2007). La intensión durante este tiempo fue poner a charlar tanto a la experiencia de aula como a la teoría de sistemas multiagentes para que se inspiraran mutuamente. Esta sección pretende dar cuenta de los momentos más relevantes de dicho diálogo, identificando los hitos más importantes durante la construcción del modelo.

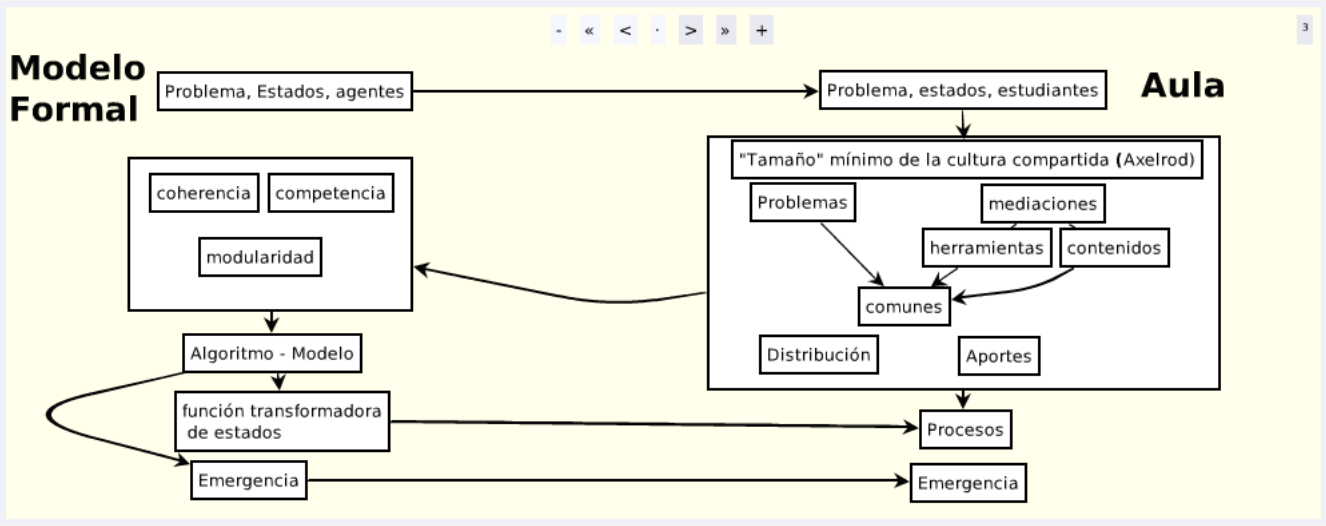

Figura 1. Esquema del diálogo entre el modelo formal y el aula

1. En este sentido se puede decir que nuestro sistema multiagente resolutor de problemas es un motor-diferencial bajo el enfoque de Sociedad de la Mente de Minsky.

Primer momento: No hay coherencia, competencia, ni modularidad

Durante el primer semestre de la experiencia, el modelo no era claro (de hecho, sólo algunas cosas eran claras y no había modelo propiamente dicho), así que se pretendió observar el surgimiento del problema colectivo a partir la configuración de una mediación digital que sirviera como un espacio común para el nacimiento de dichos problemas. Para esto se configuró y puso a disposición de los estudiantes y profesores del curso de Introducción a la Informática, un sistema colaborativo y con memoria de creación de documentos hipertextuales, llamado Eduwiki[2]. Se les solicitaba a los estudiantes que escogieran un problema de su interés para la realización durante el curso, que fueran publicando una bitácora de avance sobre ese proyecto y que leyeran y aportaran a las bitácoras de sus compañeros y se esperaba que, en la medida en que problemas distintos involucraran conceptos algorítmicos iguales (ciclos, condicionales, etc.) empezaran a surgir problemas colectivos sobre tales conceptos, cuya raíz eran los problemas particulares y diferentes de cada estudiante.

Este esquema no tuvo éxito. Los estudiantes no se leían entre sí, ni aportan consistentemente a las páginas de sus compañeros, por lo cual el problema colectivo no surgía. A esto se aunaba el hecho de que, al ser introducción a la informática una materia electiva y de matrícula abierta, se contaba con estudiantes de distintas carreras (ingeniería de sistemas, nutrición, biología, informática matemática) y distintos semestres, con preocupaciones diferentes que se reflejaban en la elección de sus problemas específicos. A pesar de que existían sugerencias sobre los lenguajes de programación y entornos a utilizar, con una predilección por los de sintaxis sencilla (Python, Smalltalk/Squeak), el hecho de que existiera tantas variantes temáticas y de implementación, aunado a una práctica en la cual no se leía a los compañeros, hizo extremadamente difícil el surgimiento de un problema colectivo. Sin embargo en este escenario surgieron las primeras inquietudes y tentativas, alimentadas por la lectura del texto La complejidad de la cooperación (Axelrod, 2004), en particular el artículo sobre difusión de la cultura

- Dado que microcultura del aula es extremadamente diversa, se necesitaba un factor unificante para la resolución colectiva de problemas y un tamaño mínimo de la cultura compartida, expresada en las prácticas culturales dentro del aula, como la publicación constante, la lectura y el comentario entre pares y las mediaciones tecnológicas de programación.

- El factor unificante no se ubicaba en la microcultura del aula sino en la macrocultura compartida, en ese sentido, un uso de los computadores profundamente influenciado por lo multimedial era un referente común en los jóvenes, en particular los estudiantes del curso, por lo que una aplicación de programación con énfasis en lo multimedial y la facilidad de uso fue escogida para todos: Squeak

- Una buena parte del trabajo extra clase tuvo que ver con ubicar literatura apropiada que ayudara en el aprendizaje de Squeak. Los tiempos asociados a preparar la experiencia y hacer una metalectura de ella fueron un factor limitánte de la pertenencia a la microcomunidad de práctica por parte del docente, lo cual no permitía encausar todo el proceso, desde adentro.

- Se empezaron a ver los primeros indicios de trabajo colaborativo y la respuesta favorable grupal por el uso de una herramienta común, de las características antes mencionadas, fue generalizada, si bien no llegamos a la solución colectiva de problemas.

- Se modificó el instrumento de evaluación de los estudiantes, para que diera cuenta de modo explícito de los valores que se querían propagar en la microcultura (pudo estar relacionado con el artículo de surgimiento de normas sociales de Axelrod en La complejidad de la cooperación, donde el instrumento de evaluación explicitaba la norma).

Segundo momento: coherencia y competencia, pero no modularidad

2. Existen muchas variantes de Wikis, en su concepción original los wikis son sólo sistemas de creación rápida de hipertexto, pero el caracter colectivo y con historia es lo que los ha hecho tan pertinentes en los escenarios de gestión de conocimiento, documentación empresarial y ha generado el éxito del proyecto Wikipedia, por ejemplo. La dirección del wiki educativo es `www.eduwiki.info`

Con la experiencia del anterior semestre se eligió Squeak desde el comienzo del siguiente semestre como un entorno de programación orientada a objetos (POO), fácil, multimedial, que hace uso de etoys, un fácil sistema de programación visual de arrastrar y soltar. Se trataba primero de adquirir el tamaño mínimo de la cultura compartida para solucionar problemas, así que después de cubrir los temas del comienzo del curso, se inició la parte introductoria a la programación, estableciendo un mínimo de actividades comunes que, si bien podrían tener ligeras variaciones de estudiante a estudiante, usaban los mismos conceptos fundamentales de la programación orientada a objetos (instanciación, agregación, polimorfismo y encapsulación) y sobre el manejo de la interface.

Las actividades de consolidación del tamaño mínimo de la cultura compartida fueron dos: la creación de un libro interactivo, usando las facilidades del bookmorph de Squeak y un juego interactivo usando los etoys. Luego de esto se procedió a resolver un problema colectivo, crear un juego, que a su vez era la resolución de un problema colectivo: sacar a un conjunto de bichos virtuales de un laberinto, mediante la asignación de funcionalidades individuales, diferentes y complementarias para cada uno de ellos (existe un popular juego de los 80's para consolas de juegos llamado Lemmings, con varios clones como Pingus; el nuestro era, en ese sentido, un clon más;). Esto permitía tener dos niveles de lectura sobre la resolución del problema: a nivel de los estudiantes y a nivel del juego mismo.

Cada estudiante debía encargarse de programar una funcionalidad específica para el bicho y deberían poder integrarse todas en un escenario virtual donde varios ejemplares del bicho (instancias en el sentido de la programación orientada a objetos) estarían encerrados. Esta segunda experiencia mostró varias cosas referidas a la interacción entre los agentes (estudiantes, bichos virtuales) y el entorno (salón-mediaciones, laberinto):

- El espacio común de interacciones entre los agentes permitía coordinar acciones entre ellos. Cuando una funcionalidad, que era pertinente a varios subproblemas, emergía, esta actuaba como atractor (en el sentido de los sistemas dinámicos complejos). Esto se veía tanto en las páginas wiki de publicación de los estudiantes, así como en la formulación de preguntas a otros estudiantes durante las sesiones del aula (se iba formando una microcomunidad de discurso[3]), y también en los códigos de mayor funcionalidad de los bichos virtuales o de los entornos que habitaban. Tanto en el caso de las páginas wiki, como de la charlas en el aula y del código en los objetos de software y sus entornos, los atractores ayudaban a organizar la actividad de resolución de problemas por parte de los agentes, bien fueran otros estudiantes en el curso (agentes naturales) o bichos de software (agentes artificiales).

- Se presentaron los primeros casos de bifurcación: soluciones alternativas al mismo problema de funcionalidad. Esto mostró una primera aproximación a la modularidad en términos del "valor de opción" (Baldwin y Clark, 2003), que precisamente tiene que ver con la posibilidad de elegir diferentes implementaciones de respuestas posibles a un mismo (sub)problema. La solución más robusta obraba como atractor en el sentido antes descrito y era elegida después de ser puesta a prueba en el aula, mediante la ejecución de su funcionalidad en el entorno para el que se había creado y de discutirse por estudiantes y profesor.

- El tamaño mínimo de la cultura compartida se puede ver en los indicadores de competencia y coherencia de los agentes individuales y a su vez estos son interpendientes de la modularidad. Como bien lo indica la literatura de sistemas multiagente (Durfee 1999), la coherencia es fácil de lograr o se presupone como una condición del sistema. En este caso se hizo un diseño de ambiente de aprendizaje que la embebiera, mediante un proyecto común con subproyectos individuales, así como instrumentos de evaluación que poderaban y alentaban la coherencia. Una vez se tiene un conocimiento compartido suficientemente grande y si el diseño del problema lo permite, surgirán las soluciones alternativas, que pueden ser intercambiadas en un entorno. En el caso de los estudiantes, era fácil ver

3. Nos referimos a este concepto desde Fish (1980), citado por Brown y otros en su escrito Conocimiento especializado distribuído en el aula (Compilado en Salomon 1993), en la que «los participantes son iniciados en los rituales del discurso y la actividad académicos y, más específicamente, científicos».

las soluciones alternativas, en el caso de los agentes de software, la coherencia se había forzado en el diseño (como en su correlato de aula) y la competencia y modularidad se podían ver en la cantidad de mensajes diferentes (métodos en POO) que estaban en condiciones de entender en común y la forma en que dichos mensajes se implementaban.

- Habíamos alcanzado un límite respecto a la modularidad y esto se debía a que el diseño de los etoys deliberadamente colocaba un límite sintáctico a las posibilidades expresivas con el lenguaje, de modo que no se colocaran complejidades extras a los niños, para las que no estaban preparados, pues eran ellos los usuarios objetivo del mismo. Sin embargo, esto restringía las posibilidades de intercambiar pequeñas partes del código que implementaban funcionalidades específicas, y teníamos que compartir objetos completos con todos sus métodos o usar funcionalidades extendidas no provistas por etoys que hacían complicado el manejo de la interface en este punto y, si bien contabamos con ayuda de miembros de la comunidad de Squeakers internacional, que implementaron algunos métodos de más bajo nivel por nosotros, pasar de la programación a través de mosaicos arrastrando y soltando, a un código sencillo sintácticamente, pero que no habíamos leído previamente era muy difícil en ese punto[4] y por tanto no podíamos modificarlo, adaptarlo o subdividirlo para garantizar la funcionalidad completa de las partes del proyecto.

Tercer momento: coherencia, compentencia y modularidad

Para el tercer semestre la experiencia se reinterpretó y rediseño a la luz de los aprendizjes de los dos semestres anteriores. Para no enfrentar inconvenientes límite sintáctico de etoys se pensó en establecer un puente entre ellos y el código en Smalltalk que está por debajo de su funcionamiento, para lo cual se usó una variante creada en Squeak, llamada Bots Inc (Ducasse, 2005), que está especialmente concebida para la enseñanza de los conceptos de programación, al mismo tiempo que presenta un correlato gráfico en los algoritmos, desde la idea de programación de robots virtuales de software que pueden dejar trazos en un entorno (en una inspiración similar a la de Logo, pero desde el entorno objetual multimedial de Squeak). Esta vez se trataba de extender la funcionalidad de los robots de modo que estuvieran en condiciones de crear un laberinto y salir de él, afrontando diferentes peligros (huecos) y premios (bonus). Para crear la cultura compartida (aumentar los índices de coherencia y compentencia), no sólo se replicó el diseño del entorno educativo en términos de los instrumentos de evaluación, las metodologías, mediaciones y contenidos (eduwiki, Squeak, Bookmorphs, eToys, etc), sino que se extendió, para incluir nuevas mediaciones y contenidos, en particular el libro y el entorno de Bots Inc, lo que permitía acceder a componentes más modulares del código e intercambiarlos sin mayor problema. Además de los atractores y bifurcaciones presentados en el caso anterior, se podrían resaltar estos momentos del correlato educativo del modelo:

- Se pasa de el tamaño mínimo de la cultura compartido, un indicador grupal o de entorno, inspirado en la terminología de Axelrod, a coherencia y competencia, un indicadores asociados a los agentes individuales. La cultura compartida del sistema tiene un correlato en la coherencia y competencia de los agentes.

- Se explicita la modularidad como la característica que no permtía intercambiar trozos de código, cuando se alcanzó el límite sintáctico de los etoys y se establece una correlación entre modularidad y coherencia: los agentes estarán dispuestos a asumir roles activos en la solución colectivo del problema, si los trozos están bien definidos, son suficientemente pequeños (en alguna métrica de tamaño) y son intercambiables.

- Se configuró un espacio de aula con coherencia (los instrumentos de evaluación, las mediaciones y las dinámicas valoraban y alentaban mucho el trabajo conjunto, los aportes individuales y la distribución), competencia (se cubrieron y evaluaron contenidos iguales hasta cierta parte del semestre) y modularidad (se diseñó un problema colectivo con subproblemas interdependientes en un entorno dónde el código fuera fácilmente intercambiable).

4. 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.

- Se empezaron a ver los primeros fenómenos emergentes. La pregunta por la emergencia en el aula fue una de las más difíciles en el correlato educativo. ¿Qué era aquello que, en términos del correlato educativo del modelo de resolución colectiva de problemas, estaba en el sistema, pero no en sus componentes individuales?. Para apreciar el fenómeno nos adscribimos a la noción de aula que proponen Brown y otros (compilado por Salomon, 1999):

  «Teóricamente concebimos el aula como compuesta por zonas de desarrollo próximo (Vigotsky, 1978) a través de las cuales los participantes pueden desplazarse por diferentes rutas y a diferentes velocidades (Brown y Reeve, 1987). Una zona de desarrollo próximo puede incluir a personas, adultos y niños, con diferentes grados de conocimiento especializado, pero puede abarcatar también artefactos tales como libros, videos, láminas murales, equipos científicos y un contexto informático destinado a apoyar el aprendizaje intencional (Campione, Brown y Jay, 1992; Scardamalia y Bereister, 1991). Una zona de desarrollo próximo es la región de actividad que los alumnos pueden recorrer con ayuda proveniente de un contexto de apoyo que incluya a personas pero no se limita a ellas (Vigotsky, 1978). Define la distancia entre los niveles reales de compresión y los que pueden alcanzarse en colaboración con personas o con artefactos poderosos. La zona de desarrollo próxima encarna un concepto de disposición a aprender que subraya niveles superiores de competencia»

  La propiedad emergente está referida precisamente a la resolución del problema, dado que esta es una capacidad del sistema, que no se encuentra individualmente en ninguno de los agentes. Por supuesto, decir que, si hay competencia y coherencia, entre todos solucionamos un problema mejor que cada uno, es una premisa básica de la cual parte la resolución de problemas en sistemas multiagente. El aporte del modelo es asociar a estas dos características de los agentes, una del problema, que es la de modularidad. Pero el énfasis del fenómeno emergencia, en términos de correlato educativo del modelo, no está allí, sino en el hecho de que, como un todo, el sistema se hace más competente (de modos no lineales) con el aumento de las competencias individuales y, a pesar de que esta competencia grupal afecta positivamente la competencia individual, una vez los agentes/estudiantes son considerados en soledad (como mostraron las evaluaciones del curso) la competencia decrece considerablemente. La emergencia se da en tanto que las zonas de desarrollo superpuestas del aula, modifican la competencia del sistema para la resolución de problemas y dejan "resagos" en las competencias individuales. La competencia grupal es un fenómeno emergente que surge del hecho de que las competencias individuales dejan huellas visibles en el entorno constituido por un problema modular.

El modelo

El sistema multiagente se caracterízará por tres elementos para la resolución de un problema:

- Coherencia (querer resolverlo): «La propiedad o estado de actuar como una unidad. La coherencia tiene que ver con qué tan bien se comporta el sistema como una unidad. Los criterios de evaluación para la coherencia son, por ejemplo, eficiencia, calidad de la solución y degradación elengante en presencia de fallas» (Huhns, Michael N. y Stephens, Larry M., 1999). Obsérvese que, si bien la coherencia es una propiedad emergente del sistema, se requiere que cada agente «quiera» actuar como unidad, de acuerdo a la labor que le fue asignada. Es decir que la coherencia grupal surge del aporte individual de los agentes.
- Competencia (saber resolverlo): «Es la habilidad de hacer una tarea bien» (Durfee, Edmund, H., 1999). La coherencia sólamente no basta para resolver un problema, sino que se requiere que cada uno de los agentes individuales esté en condiciones de resolver la parte del problema que le fue encomendada. De nuevo, se trata de una propiedad global que surge de los aportes individuales de los agentes.

- Modularidad (descomponible en partes acopladas): «Es referida como una propiedad general de los sistemas complejos, concerniente al grado de descomponibilidad del sistema en subpartes vagamente acopladas hechas componentes fuertemente acoplados. [...] Un sistema modular, entonces, es representado como un complejo de componentes o sub-sistemas donde los diseñadores tratan de minimizar y estandarizar las interdepenencias entre los módulos» (Schilling, 2000; citado por Narduzzo y Rossi).

Obsérvese que, mientras las dos primeras características están referidas a los agentes individuales, la última se refiere al problema, que en este caso es modelado por el entorno de los agentes, y por tanto también es una característica referida a la relación e interacción entre los agentes y el problema.

Es de anotar también que los agentes no habitan un espacio físico, sino un espacio formal, donde se modela el problema. El entorno y los agentes admiten representaciones gráficas bidimensionales a fin de visualizar el modelo, sin embargo se intenta capturar las características genéricas de un problema en un espacio formal y de los agentes que lo han de resolver.

Las características de competencia y coherencia, si bien son poseídas individualmente por los agentes, pueden ser leídas desde premisas asociadas a ellos como pertenecientes a un colectivo. Es decir, determinados niveles de coherencia y compentencia individuales, pueden dar cuenta de un conjunto de valores o ethos y saberes comunes, que a su vez pueden dar cuenta de fenómenos complejos, como el tamaño de la cultura compartida[5]. Los agentes, entonces, tendrán indices de coherencia y competencia que podrían dar cuenta de que todos tienen intensiones o saberes similares, o bien, que a pesar de ser saberes distintos, son complementarios en la resolución del problema. El hecho de que los agentes se comuniquen a través de las percepciones y modificaciones del entorno también da cuenta de fenómenos presentes en el sistema, más que en los agentes individuales, es decir asociadas a la interconexión e interdependencia, que a su vez podrían ser asociadas a propiedades no explícitas en el modelo, como las culturales.

Conservaremos acá la definición de agente como una función transformadora, presentada anteriormente, pero la afinaremos para dar cuenta de las características de coherencia, competencia y modularidad. Decíamos que un agente era una función

$$Ag: \mathcal{R}^E \rightarrow Ac$$

de $\mathcal{R}^E$, el subconjunto de las ejecuciones $\mathcal{R}$, cuyas ejecuciones terminan en estados, en el conjunto de acciones Ac. Esta definición presentaba el dominio y el rango de la función, pero no la definía de modo explícito (puesto que era una definición general). Para efectos del modelo, se procederá a definirla acá de forma específica a partir de cómo el agente «tomará una decisión» sobre si cambiar o no el entorno a partir de sus niveles de coherencia y competencia.

Este es el algoritmo del agente para cada ciclo de ejecución:

- Tamaño del problema > 0 ?
    - Si:
        - Nivel de coherencia está por encima de un mínimo prefijado?
            - Si:
                - Es competente para aportar a la solución de ese módulo del problema?
                    - Si:
                        - Aporta a la solución cambiando el valor del estado del entorno en donde se encuentra ubicado.

5. Suponemos acá un acercamiento similar al que hace Axelrod (2004) para la cultura, es decir, la idea de que para efectos de un sistema multiagente, las cultura puede ser modelada como un conjunto discreto de variables que a su vez toma un conjunto discreto de valores. Dos agentes poseen la misma cultura si la misma variable tiene el mismo valor. Es de anotar, sin embargo, que el modelo presentado en este trabajo no coloca a la cultura en una estructura de datos de los agentes y/o del entorno de modo explícito, sino que la modela implícitamente a patir de los niveles de coherencia y competencia de los agentes, así como de modularidad del problema.

- Disminuye el tamaño del problema en una unidad.

– No:

- Se desplaza a otro «lugar» del estado del entorno.

• No:

◦ Se desplaza a otro «lugar» del estado del entorno.

◦ No

– Se detiene

Implementación computacional

Para la implementación del modelo se escogió Kedama (Yoshiki, sin fecha), un sistema de simulación multiagente, implementado en Squeak, que se ha empleado para realizar modelos con agentes reactivos, ya sean estos biológicos, como en las colonias de hormigas (Yoshiki, sin fecha), o sociales, como la segregación de Schelling (Guillaume, 2006), entre otros. Las razones de su escogencia no sólo están asociadas a su adecuación para la simulación que se pretende, sino, además, al hecho de brindar un sistema de programación visual vía mosaicos (eToys), de poseer un lenguaje subyacente orientado a objetos de sintaxis simple y elegante y de ser Squeak el sistema con el cual se estableció el correlato y la experiencia de aula y educativa del modelo, cuya implementación se muestra en esta sección.

Kedama está hecho de varias partes. En la parte inferior está el mundo Kedama en sí mismo, el cual aparece como una gran caja negra. Encima de estos está una grilla invisible de cuadrados. Cada cuadrado es llamado un patch (parche), y la grilla entera es controlada por variables parche que se pueden programar. Los parches sólo son encontrados en Kedama y pueden ser pensados como de un papel de gráficos con 100 x 100 celdas (10.000 en total). A cada celda puede ser asignado un valor numérico. Moviéndose a través de esos parches en el mundo Kedama están los rebaños de tortugas, que son grupos de tortugas que actuan en la misma forma. Estas tortugas, moviendose a lo largo del mundo Kedama, pueden también leer el valor de cada parche y responder a él. Finalmente, hay tortugas individuales, o partículas, que comparten en mismo nombre que su rebaño. Por ejemplo, si el primer rebaño que se crea es `tortuga1`, cada tortuga en este rebaño seguirá las instrucciones a la `tortuga1`. A las tortugas se les da las instrucciones como rebaño, no como tortugas individuales.

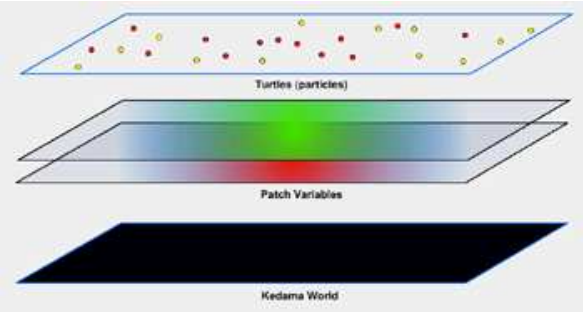

Figura 2. Arquitectura de capas del mundo Kedama

Se colocaron dos mundos Kedama, llamados `Kedama` y `Kedama1` que representan el estado final y el estado original en el que se encuentra el entorno que habitan los agente. Como se dijo en el modelo formal, la diferencia en estos estados es lo que constituye un problema. A su vez cada problema puede ser dividido en subproblemas, que en el mundo Kedama son representado por parches (patchs). Para efectos de esta implementación se colocó un problema que constaba a su vez de tres subproblemas o módulos. Se eligió una métrica sencilla para la modularidad diciendo que la modularidad es igual al número de módulos.

Se definieron tres variables en el mundo que representa el estado objetivo: `problemaTotal`, `problemaActual` y `solucionActual`, el primero mide el tamaño del problema, es decir la cantidad de valores del estado objetivo que se desean que sean diferentes a las del estado actual, el segundo el tamaño del problema cuando se está planteando, y el tercero mide el tamaño de la solución, mientras se ejecuta el algoritmo.

Se programaron dos tipos de agentes (tortugas, en la terminología de Kedama), unos que recorrían el mundo planteando diferentes tipos de problemas y otros que los solucionaban. El algoritmo método principal del agente que plantea el problema se muestra a continuación:

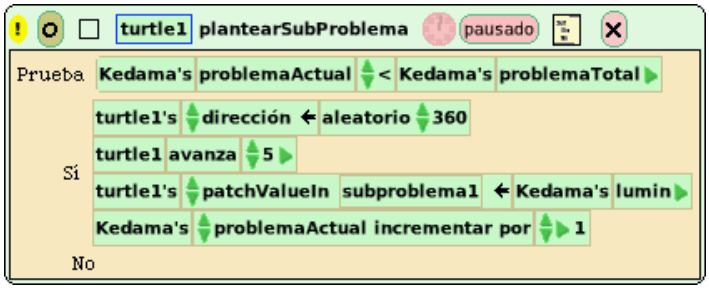

Figura 3. Método de los agentes que plantean el problema

Su funcionamiento es como sigue: El primer tipo agente compara si el problema que el tiene que ayudar a plantear ya es del tamaño del problema deseado, en caso de que no lo sea (es decir que `problemaActual` sea menor que `problemaTotal`) elije un lugar aleatorio dónde ubicarse y escribe en el patch `subproblema1`, que representa un módulo del problema que está ayudando a plantear, y cambia el valor del patch, incrementando luego el `problemaActual` en una unidad. De modo análogo ocurre con los otros dos tipos agentes que plantean los otros subproblemas. Nótese que aquí hemos hablado de tipos de agentes y no de agentes específicos, dado que podemos colocar a varios agentes de un sólo tipo a que ayuden a plantear un subproblema, caso en el cual lo valores diferentes entre el estado inicial del entorno y el estado objetivo estarán más exparcidos por todo el entorno.

La resolución del problema es la que implementa en Kedama el algoritmo de la sección anterior y el núcleo de esta tesis. Se han colocado dos tipos de agentes para solucionar los tres subproblemas, unos, los verdes, que sólo puede solucionar subproblemas del tipo 3 (o planteados en espacio del patch 3) y otros, los rojos que puede solucionar problemas del tipo 1 o 2 (planteados en los espacios de los patch 1 y 2). En la terminología del modelo los agentes rojos son más competentes que los agentes verdes. Siguiendo la hipótesis de Narduzzo y Rossi (2003) en cuanto a que problemas más modulares afectan positivamente el deseo de ser solucionados por los agentes en un colectivo (pues se pueden encargar de secciones más pequeñas), se ha puesto un factor de aleatoriedad, que modela la coherencia y se relaciona con la modularidad de este modo: si el número aleatorio que modela la coherencia es menor que la modularidad entonces el agente intentará verificar su competencia (si puede aplicar su saber a la solución) y en caso de que sea competente y el valor en el espacio solución aún esté sin solucionar, aportará a la solución cambiando el estado en el patch que modela la subsolución e indicando esto mediante el incremento de la variable `solucionActual` en una unidad. El algoritmo, implementado en Kedama, es como sigue:

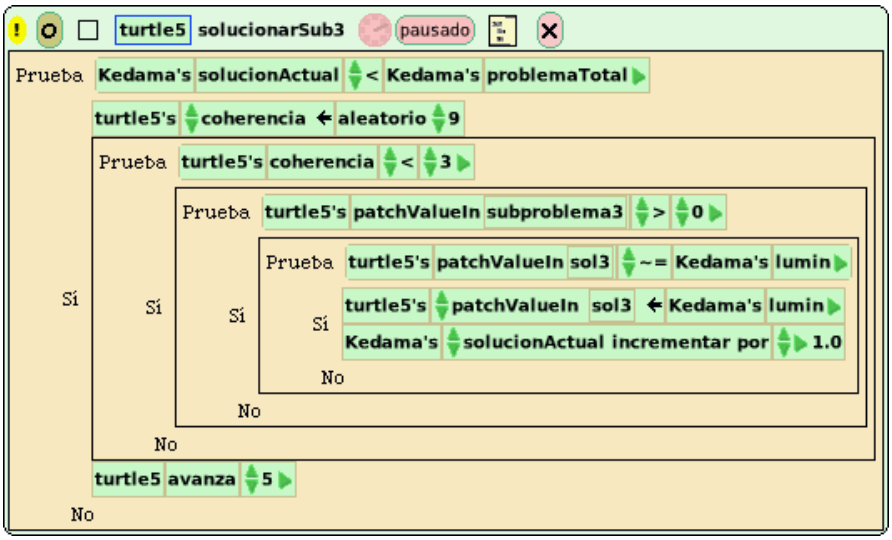

Figura 4. Algoritmo de solución de problemas del modelo presentado en esta tesis implementado en Kedama. Se deben usar pruebas anidadas para simular el operardor "y" lógico, pues el sistema de programación gráfico eToys de la versión actual de Kedama no lo soporta en la programación gráfica que se emplea.

El algoritmo de los agentes rojos es similar en esencia, con la diferencia de que, al ser más competentes, deben hacer más verificaciones anidadas, asociadas a dónde pueden aplicar su saber (si en el subproblema uno y/ó en el subproblema 2). La implementación en Kedama se muestra a continuación:

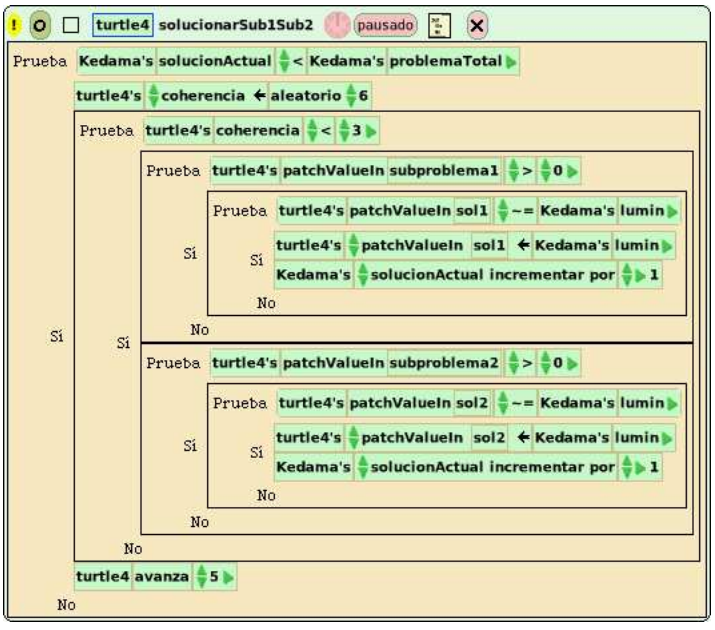

Figura 5. Implementación del algoritmo del modelo, para agentes más competentes

Finalmente se implementó en en entorno Squeak una interface con un conjunto de controles que permitieran correr varias simulaciones cómodamente, especificando el tamaño del parámetro `problemaTotal`, y controlar los momentos en los cuales se invocaban los métodos de planteamiento de problemas y ejecución de soluciones por parte de las familias de agentes. Unas captura de pantalla del entorno se muestran a continuación:

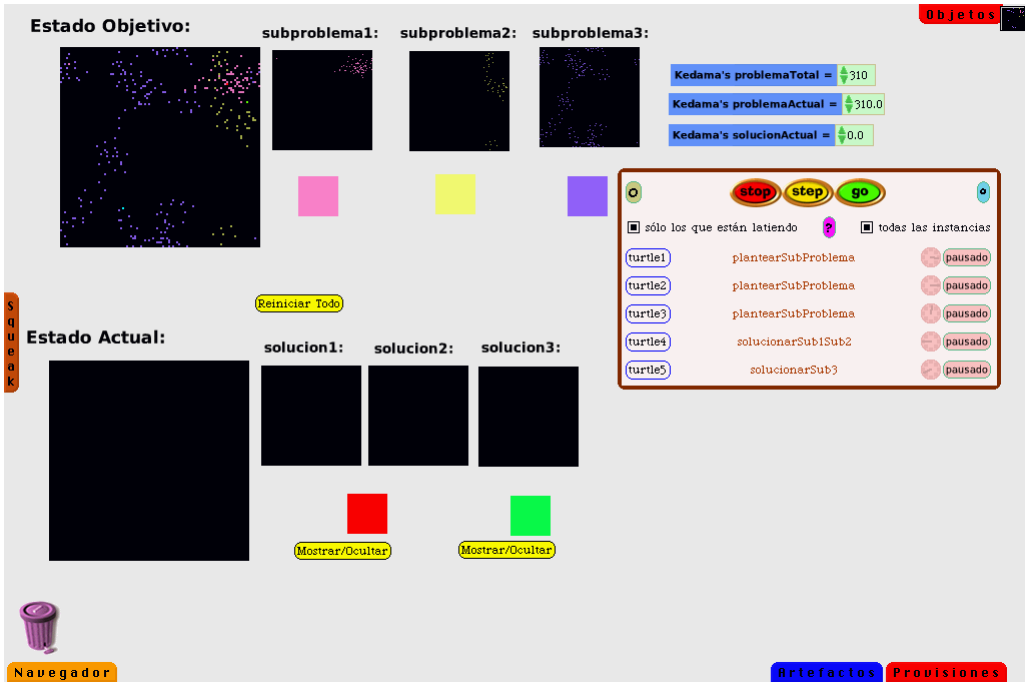

Figura 6. El entorno de ejecución de las simulaciones del modelo. Se puede ver el estado objetivo del entorno, constituido a su vez por tres subproblemas o módulos y, abajo de cada uno, ejemplares de las familias de agentes que ayudan a plantearlos. A la derecha están los valores de las variables más importantes del entorno y una utilidas que permite invocar la ejecución individual de los métodos para plantear subproblemas o sus soluciones. En la parte inferior se puede ver el estado actual del entorno, antes de invocar la solución del problema, junto con las subsoluciones y los ejemplares que las resuelven.

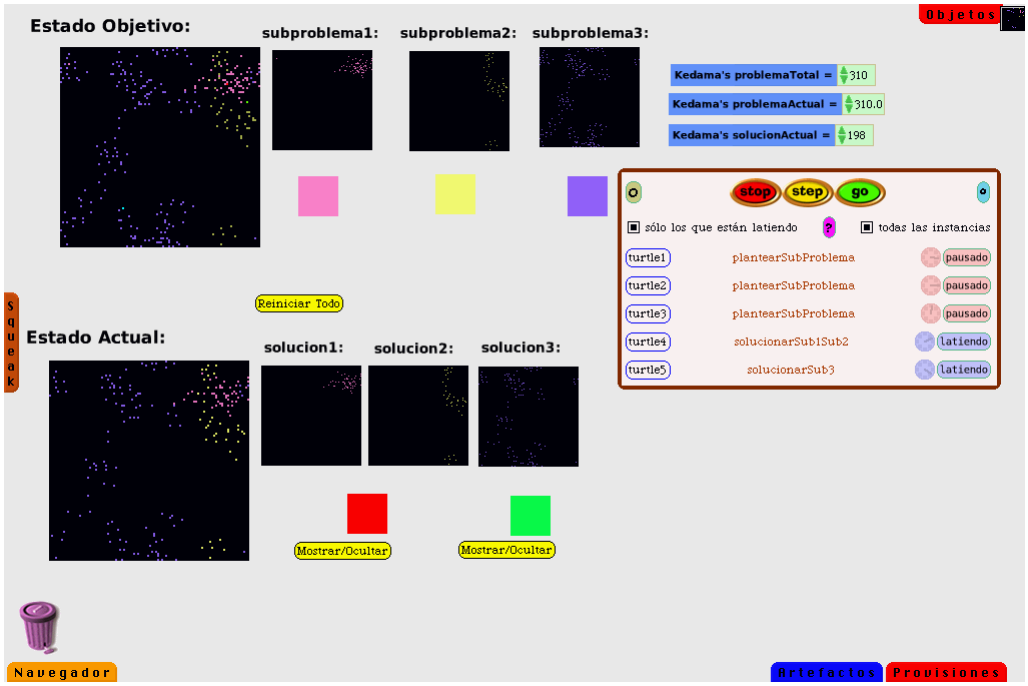

Figura 7. Simulación ejecutando los algoritmos de resolución del modelo, lo que se aprecia porque los relojitos que controlan a las familias de agentes rojo y verde tienen la leyenda "latiendo" en la utilidad de control de guiones a la derecha y están de color azul. Obsérvese cómo las soluciones 1 a 3 de la parte inferior empiezan a parecerse a los subproblemas del estado objetivo en la parte superior.

Conclusiones y recomendaciones

La resolución colectiva de problemas puede ser caracterizada efectivamente, de modo general, a partir de tres propiedades: coherencia, competencia y modularidad. Las dos primeras son propiedades de los agentes, mientras que la última es una propiedad del problema (entorno). Este es, hasta donde la revisión literaria permitió apreciarlo, el primer modelo que tiene en cuenta las tres propiedades en conjunto. La coherencia y la modularidad pueden ser vistas en términos de una propiedad del sistema, relacionado con el tamaño mínimo de la cultura compartida, desde el modelo cultura dentro de la perspectiva de Axelrod 2004, es decir, como un conjunto de variables que poseen unos valores. Cuando las variables tienen los mismos valores se habla de regiones del cultura y ellas implican, en términos de coherencia y competencia, intenciones y vocabularios compartidos por los agentes, los cuales a su vez se emplean y explicitan en constructos más complejos (métodos y mensajes de programación orientada a objetos, foros wiki, etc.).

Microcambios generan macrocomportamientos. En este sentido se valida lo expuesto por Schelling en su modelo de segregación, aplicándolo también a la resolución colectiva de problemas. Esto se explicita en cómo la experiencia de aula intento propagar un ethos y un conocimiento comunes en los estudiantes individualmente a partir de instrumentos de auto y heteroevaluación y como se consideraba la publicación y el uso individual de las mediaciones, a pesar de que pretendía finalmente evaluar un proyecto colectivo, sin descuidar por ello los procesos y aportes individuales y a los subproblemas. El fenómeno emergente de la resolución colectiva de problemas, que estaba por encima de las capacidades individuales de cada uno de los agentes fue una de las evidencias del macrocambio. Estos microcambios ocurren porque cada uno de los agentes incorpora las intenciones y los vocabularios como reglas permanentes de su interacción con el entorno (como en el caso de la simulación de la colonia de hormigas) o bien porque los agentes se propagan, en su interacción, las reglas entre sí (como en la simulación de la epidemia). Sería interesante ahondar más en el correlato educativo de esta conclusión desde las ideas de zonas de desarrollo próxima y su jalonamiento en la relación estudiante-estudiante y estudiante-entorno y la metáfora empleada del aula como un espacio donde se superponen múltiples zonas de desarrollo.

Entre más cerca se está a hallar la solución ideal más tiempo tardan los agentes en realizar aportes a dicha solución. Este fue uno de los fenómenos emergentes más interesantes al correr la simulación computacional del modelo repetidas ocasiones. Debido a que los agentes ya han aportado los saberes en su recorrido por el espacio de búsqueda, ahora se trata de pasar por aquel lugar específico del espacio donde aún el aporte no se ha hecho, sin embargo mucha de la exploración ocurre en donde ya se ha aportado la solución, que es un lugar más grande que el lugar donde el aporte específico falta. Efectivamente en el correlato de aula, una vez habíamos integrado gran parte del prototipo del proyecto colectivo, la depuración de errores específicos y la resolución de codependencias no previstas tomó mucho más tiempo que la integración original y de hecho no se alcanzó a realizar del todo, debido a los costos que tiene lograr el tamaño mínimo de la cultura compartida. Es importante, por esto, una vez logradas coherencia y competencia en el sistema, ofrecer correlatos curriculares que le permitan al sistema continuar existiendo y funcionando para aportes más específicos a (sub)problemas puntuales.

Narduzzo y Rossi (2003) muestran como la modularidad es una característica arquitectónica que, en el caso específico del desarrollo de software, mejora las posibilidades de que hayan contribuciones. En términos más generales del modelo acá presentado, podríamos decir que la modularidad del problema afecta positivamente la coherencia de los agentes. Esto se vió claramente en el correlato de aula, cuando, una vez alcanzado el límite sintáctico provisto por los eToys, fue muy difícil realizar la integración de los subproblemas individuales que hacían parte del problema general, mientras que usando Bots Inc la posibilidad de deconstrucción e integración aumento y al final se logró resolver el problema colectivo planteado.

El aumento progresivo de la competencia se ve favorecido por metáforas deconstruibles y continuas. Los conceptos básicos de la programación orientada a objetos (herecia, encapsulación, polimorfismo y agregación) se abordaron varias veces a través de mediaciones y actividades distintas pero continuas y complementarias: al principio se ofreció la construcción de un

Bookmorph interactivo (para establecer continuidad con actividades previas como el uso de presentaciónes en PowerPoint u OpenImpress), luego los mismos elementos básicos se emplearon en la construcción de un juego interactivo para enfatizar los componentes de programación y encontrar el límite de los eToy y posteriormente se pasó Bots Inc donde se mantenía un correlato gráfico de los conceptos algorítmicos y de programación orientada a objetos, pero se tenía acceso a constructos sintácticos mucho más simples, de bajo nivel y poderosos. Lo deconstruible de la metáfora tiene que ver con los aumentos de la competencia del agente cognitivo y gracias a dichos aumentos se incrementa la modularidad y el valor de opción dentro de las soluciones al problema.

Las codependencias no previstas son parte de un problema de descomposición (Narduzzo y Rossi 2003) y es bueno tener funciones integradoras que ayuden a resolverlos (por ejemplo, el "macro" método que invocaba los métodos particulares en el correlato de aula). Alguna de las funcionalidades de los módulos-métodos que resolvían problemas específicos al ser integradas en el problema general mostraban codependencias no previstas entre los subproblemas que afectaban la integración y en este caso lo que se hizo fue relegar comportamientos de los métodos específicos e integrar las partes que resolvían las codependencias al método general. En resumen, las codependencias no previstas en las soluciones a los subproblemas pueden ser resueltas en un relegando funcionalidad conflictiva de la solución a los subproblemas e integrándola en un mecanismo de comunicación entre ellos.

Los mecanismos externos de representación y la competencia del sistema son interdependientes. En la medida en que se incrementa la competencia del sistema se hacen más complejos, densos y frecuentemente usados. En el caso de Eduwiki se pudo evidenciar cómo se pasaba de páginas personales sin comentarios, a páginas con comentarios superficiales y luego a la constitución de un foro virtual, donde se compartían avances frecuentes del proyecto común, respuestas generales a los inconvenientes de integración y unos grupos ayudaban a otros en la integración, del mismo modo se empezaron a usar sistemas de representación externos como Internet Archive, que permitía compartir todo el estado del sistema Bots Inc, cuando la integración lo requirío.

Algunas actividades en las que el sistema educativo ha favorecido usualmente el trabajo individual pueden ser reinterpretadas y beneficiadas a la luz de la cognición distribuida, por ejemplo la resolución de problemas en general y en particular la programación de computadores (que se ha visto como una actividad que realiza y aprende un programador en "solitario" frente a su máquina). En ese sentido este trabajo se suscribe a lo dicho por Dillebourg (1996), en cuanto a que las principales funciones del entorno de aprendizaje (entre ellas el el diágnostico, la tutoria y la explicación) son principalmente distribuidas y aporta modelos y mediaciones que explicitan cómo lo es en particular la solución de problemas. También se comparte la visión de que esto tiene consecuencias en el diseño de los sistemas educativos y, de hecho, el correlato de aula del modelo mostró un posible diseño que se puede emplear a fin de evidenciar y favorecer la resolución colectiva de problemas.

Diseños de aula y evaluación pueden favorecer la coherencia y la competencia aumentando la conciencia del sistema sobre sí mismo (emergencia fuerte). El instrumento de evaluación utilizado durante las experiencias de aula también evolucionó y tuvo un caracter diagnóstico y alentador de competencia individual y coherencia (la evaluación dinámica favorece la coherencia y la competencia de los agentes). Lo primero porque si un saber ya evaluado mejoraba, la nota en el instrumento lo reflejaba y lo segundo porque un elemento permanente de evaluación eran los comentarios hechos a las páginas de otros y la integración del trabajo individual en el proyecto común, de modo que se alentaba a leerse entre sí y a aportar soluciones conjuntas o a otros.

Pueden haber correlatos curriculares que tenga que ver con saberes comunes que luego toman subproblemas distintos y afectan la experticia de quienes los tienen o usan/modifican la preexistente. En este sentido se pueden repensar los currículos de modo tal que los cursos ofrezcan contenidos curriculares similares (que configuran el tamaño mínimo de la cultura compartida), pero también niveles de especialización al interior de ellos, de modo que se puedan generar variaciones sobre esos saberes comunes, aplicados a subproblemas específicos o que permitan usar otros saberes en los subproblemas. Esto ayuda a la configuración de una comunidad de discurso y práctica pues genera equilibrio entre los saberes comunes y los complementarios.

Posteriores estudios sobre este mismo tema podrían mejorar el modelo computacional y las aplicaciones y diálogos con el correlato educativo, en particular en lo referido a diseño de redes de aprendices que solucionen problemas colectivamente y a las mediaciones que las favorezcan, enfatizando dos aspectos: el paso del agente individual a la agencia colectiva y la validación no sólo funcional, sino también comportamental del modelo o las modificaciones al mismo a la luz de ser más cercano a dichas validaciones y explicitar aún más los alcances y las limitaciones del modelo.

Bibliografía

- 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