Incorporación de Agentes Inteligentes en Ambientes de Aprendizaje

Gabriel Jaime Gómez B, Claudia María Zea R.
Proyecto CONEXIONES.
Universidad EAFIT, Medellín - Colombia.
e-mail:gabrigom@conexred.eafit.edu.co, czea@conexred.eafit.edu.co

Resumen

Este artículo describe el diseño de Agentes Inteligentes y la experimentación con estos en ambientes de aprendizaje, haciendo uso de nuevas tecnologías tales como el Lenguaje para la manipulación de Consultas y Conocimiento (KQML), y características nuevas del Lenguaje de programación Java para la creación de aplicaciones distribuidas, tales como Invocación Remota de Métodos (RMI), programación multihilos y conectividad a Bases de Datos, JDBC.

1. Introducción.

El ambiente computacional en el que los programas del futuro estarán inmersos, se caracterizará por ser altamente distribuido, heterogéneo, extremadamente dinámico y comprenderá un gran número de nodos autónomos. Un sistema de información operando en tal ambiente, deberá ser capaz de manejar varios problemas:

- El modelo cliente servidor, predominante en Internet es muy restrictivo, por lo tanto es dificil para los servicios de información actuales, tomar la iniciativa y brindar nuevo material de interés a los usuarios. De igual forma sería deseable que algunos nodos actuaran como clientes o como servidores, dependiendo de quienes interactuan con ellos.

- Varias formas de heterogeneidad necesitan ser manejadas, por ejemplo: diferentes plataformas, diferentes formatos de datos, diferentes idiomas, diferentes servicios de información, así como las diferentes tecnologías de implementación empleadas.

- Muchas tecnologías de software tales como simulación de eventos, procesamiento de lenguaje natural aplicado, razonamiento basado en conocimiento, recuperación de información avanzada, procesamiento paralelo, están disponibles para participar y contribuir a los sistemas de información nuevos. Sin embargo, se carece de técnicas para construir clientes y servidores inteligentes , o en general, software basado en agentes.

Una sociedad de agentes inteligentes podría solucionar los problemas mencionados. Estos agentes tendrían la posibilidad de migrar de un nodo a otro, la capacidad de comunicarse con otros agentes, usando un lenguaje de comunicación expresivo, trabajar juntos cooperativamente para lograr objetivos complejos a favor de un usuario, actuar con su propia iniciativa y usar información local y conocimiento para manejar recursos y requerimientos de otros agentes.

La mayoría del software existente ignora el hecho de que generalmente las tareas que realiza un individuo forman parte de actividades colectivas. No es trabajo individual, es trabajo con aportes de terceros y consensos con otras personas, es por esto que los sistemas de apoyo al trabajo colaborativo se presentan como una necesidad marcada para el desarrollo de software futuro y los agentes inteligentes, como un componente esencial en su diseño e implementación.

El objetivo es extractar a nivel teórico la esencia de la agenticidad, diferenciar los programas basados en agentes de los programas normales y establecer un grupo de principios a tener en cuenta para el diseño de agentes y a nivel práctico presentar una interfaz para trabajo colaborativo infantil que actualmente se encuentra en construcción para su posterior utilización en las instituciones educativas asociadas al Proyecto CONEXIONES.

2. Agentes de Software: Historia.

Los agentes de software hacen parte de los sistemas multiagentes(multi-agent systems MAS), el cual a su vez es uno de tres grandes campos de la Inteligencia Artificial Distribuida (DAI). Los otros dos campos son Solución de problemas distribuida (Distributed Problem Solving, DPS) e Inteligencia Artificial Paralela (Parallel AI, PAI). Por lo tanto, los agentes de software, heredan muchas de las motivaciones, objetivos y beneficios potenciales de la Inteligencia Artificial Distribuida como: la modularidad, la velocidad propia del paralelismo, y la precisión debida a la redundancia. También heredan de la Inteligencia Artificial la operación al nivel del conocimiento y la facilidad de mantenimiento. (Huhns & Singh, 1994).

El concepto de agente se remonta a los primeros días de investigación en Inteligencia Artificial, en la época de los 70's. Carl Hewittis en 1977 propuso el concepto de un objeto auto- contenido, interactivo y de ejecución concurrente llamado actor. Este objeto tenía algún estado interno encapsulado y podría responder a mensajes de otros objetos similares. "Es un agente computacional, que tiene una dirección de correo electrónico y un comportamiento. Los actores se comunican por paso de mensajes y desarrollan sus acciones concurrentemente" (Hewitt, 1977, pág 131).

Podemos dividir la investigación en agentes en dos ramas principales, la primera en el periodo comprendido entre 1977 hasta 1990 y la segunda desde 1990 hasta nuestros días. En la primera las investigaciones se concentraron principalmente en tipos de agentes repartidores, con modelos simbólicos internos, también en aspectos de discusión macros como teoría de arquitectura, teoría de lenguaje, la interacción y la comunicación entre agentes, la descomposición y la distribución de tareas, coordinación y cooperación, resolución de conflictos vía negociación, etc. su objetivo fue especificar, el análisis y el diseño de sistemas integrados incluyendo múltiples agentes colaborativos. Estos puntos de agentes, enfatizaban en las sociedades de agentes más que en agentes individuales.

Sin embargo, a partir de los años noventa, adicionalmente a la investigación en aspectos macros, tales como la arquitectura y el lenguaje, se hace evidente e inconfundible, que el rango de los tipos de agentes siendo investigados y desarrollados es ahora mucho más amplio e incluyen una preocupación por la integración de bases de datos y sistemas de inferencia.

3. ¿Que es un agente?

Es difícil definir precisamente qué son los agentes y hay por lo menos dos razones para ello. La primera es que los investigadores en agentes inteligentes, no son dueños del término agente, ya que este es usado cotidianamente por ejemplo al decir agentes de viaje. Segundo, el término ha sido usado en la comunidad de software en campos de investigación muy heterogéneos.

Podríamos decir entonces, que la palabra agente es un meta-término o una clase, de la que muchos investigadores a falta de una definición han creado sinónimos añadiendo algo de confusión. Claro que hay buenas razones para tener tales sinónimos. Primero los agentes vienen en diferentes apariencias: por ejemplo los que habitan en un mundo físico, como una fábrica, son llamados robots, los que habitan en una vasta red de computadoras son referidos como softbots (software robots), aquellos que realizan tareas específicas son llamados taskbots (Task based robots) y autonomous agents se refiere típicamente a agentes móviles o robots que operan en ambientes dinámicos e inciertos. Segundo, los agentes pueden jugar roles específicos, por lo tanto tenemos asistentes personales o Knowbots (Knowledge-based robots), que pueden tener el conocimiento de un experto en un dominio específico del saber humano.

Además, debido a la multiplicidad de roles que los agentes pueden jugar, hay ahora una plétora de adjetivos que califican a la palabra "agente": agentes de navegación, agentes de búsqueda y recuperación de información, agentes de dominio específico, agentes de desarrollo, agentes de análisis y diseño, agentes para pruebas, agentes de ayuda, etc.

La esencia de la agenticidad

Ejemplos de agentes son los humanos, animales, algunos robots móviles. Todos ellos actúan en el mundo real, los agentes de software "viven" en sistemas operativos de computadoras, bases de datos, redes, etc. Casi todas las definiciones anteriores se refieren a agentes de software. Finalmente agentes con vida artificial "viven" en ambientes artificiales en una pantalla de computadora o en la memoria de ésta.

¿Qué comparten estos agentes que constituya la esencia de ser agente? Cada uno de ellos está situado en y es parte de algún ambiente. Cada uno muestra su ambiente y actúa autónomamente sobre éste. No requieren de otra entidad para alimentarse de entradas, o para interpretar éstas y usar su salida. Cada uno actúa en pos de su propia agenda, así sea satisfaciendo conductas dirigidas como los humanos y los animales, o persiguiendo objetivos desarrollados por algún otro agente, como los agentes de software. Cada acción que realizan puede afectar su posterior sesitividad, ya que sus acciones afectan el ambiente.

Finalmente, cada uno actúa continuamente sobre algún periodo de tiempo. Un agente de software, una vez invocado, típicamente corre hasta que decide no hacerlo. Un agente con vida artificial a menudo corre hasta ser devorado o hasta que resulta muerto de otra manera. Por supuesto algún humano puede tirar la conexión, pero no siempre.

Para nosotros estos requerimientos constituyen la esencia de ser de un agente. Formalicemos la definición: *Un agente autónomo* es un sistema situado dentro de un ambiente y como parte de ese ambiente lo muestra y actúa sobre él en el tiempo, en pro de su propia agenda de actividades y así afecta lo que censará en el futuro.

4. ¿Que pasa con los programas ordinarios?

Un programa de nómina en un ambiente del mundo real podría decirse que muestra el mundo vía su entrada y actúa en él vía su salida, pero no es un agente porque su salida normalmente no afectaría lo que se muestra con posterioridad. Un programa de nómina también falla el test de continuidad temporal. Éste corre una vez y entonces entra en coma, esperando ser llamado nuevamente. La mayoría de los programas ordinarios son excluidos por una o ambas de estas condiciones, sin mucha consideración de como restrinjamos el ambiente. Podríamos decir entonces, que todos los agentes de software son programas, pero no todos los programas son agentes.

Clasificación de los agentes

Como se pudo observar anteriormente, lograr tener una única definición de agente es complicado, también lo es clasificarlos debido a que en esencia, los agentes existen en un ambiente multidimensional, por lo que usar una matriz de dos o tres dimensiones para clasificarlos, sería incompleto e inadecuado, por lo tanto exploremos otros esquemas.

5.1 Una taxonomía natural de clases de agentes:

A semejanza del modelo biológico "Al nivel de Reino" podríamos clasificar los agentes autónomos como biológicos, robóticos o computacionales. Estas parecen ser clases naturales[Keil, 1989]. En toda cultura, aún los niños muy pequeños fácilmente distinguen entre organismos animados, artefactos y conceptos abstractos.
Al nivel Phylum podemos subclasificar razonablemente los agentes computacionales en agentes de software y agentes con vida artificial.

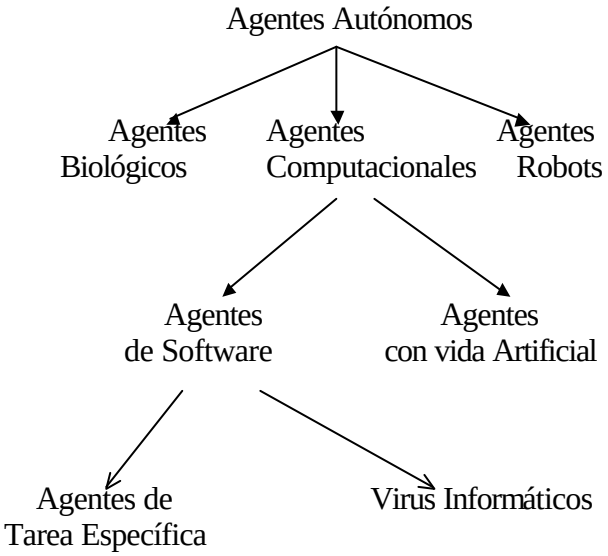

Figura 1.

Según las estructuras de control

Según las estructuras de control, la taxonomía de agentes de software de Bustoloni[1991] comienza con una clasificación de tres vías en:

Agentes reflejos que se limitan a clasificar las percepciones recibidas, produciendo la respuesta respectiva, reaccionando a cada entrada sensorial, y que siempre conocen qué hacer, es decir que no planean ni aprenden.

Agentes que planean utilizando cualquiera de los mecanismos usuales de razonamiento de la IA, o utilizando métodos de investigación de operaciones (OR agents), o utilizando algoritmos aleatorios (randomizing agents), redes neuronales, aprendizaje de máquina, etc.

Agentes adaptativos, los cuales no únicamente planean, sino que también pueden aprender.

Según el ambiente en el que habitan

La relación que existe entre los agentes y los ambientes es siempre la misma: es el agente quien ejerce sobre el ambiente, el que a su vez, aporta percepciones al agente. Los ambientes son de diversos tipos y condicionan el diseño de los agentes. Puede decirse que los ambientes son:

- Accesibles o no accesibles
- Deterministas o no deterministas
- Episódicos o no episódicos
- Estáticos o dinámicos
- Discretos o continuos

El caso más difícil se caracteriza por ser inaccesible, no episódico, dinámico y continuo.

Según sus atributos

A continuación definimos algunos de los principales atributos que pueden tener los agentes:

- Reactivo : El agente responde en un determinado tiempo a cambios en el ambiente.

- Autónomo: El agente no simplemente actúa como respuesta a estímulos en el ambiente, sino que exhibe un comportamiento dirigido a lograr sus objetivos.
Un elemento clave de esta autonomía es su proactividad, su, habilidad para "tomar la iniciativa" mas que actuar simplemente en respuesta a su ambiente.

- Continuidad Temporal : El agente está continuamente en busca de sus objetivos.

- Comunicativo: El agente es capaz de comunicarse con otros agentes, tal vez con personas.

- Aprender: La clave para lograr algún nivel de inteligencia, es la habilidad de aprender. El aprendizaje puede también tomar la forma de desarrollo incremental en el tiempo. El agente es capaz de realizar cambios en su ambiente basados en su experiencia previa.

- Movilidad: Capaz de transportarse a sí mismo de un lugar a otro

- Flexibilidad: El agente es capaz de adaptarse a cambios inesperados en el ambiente.

- Cooperación: En orden a cooperar, los agentes necesitan poseer una habilidad social, por ejemplo para interactuar con otros agentes y posiblemente humanos vía algún lenguaje de comunicación.

- Carácter: Creíble personalidad y estado emocional

Consideramos que hay como mínimo tres atributos primarios que idealmente deberían exhibir: autonomía, aprendizaje y cooperación con otros agentes. Usando estas tres características mínimas podemos derivar cuatro tipos de agentes: agentes colaborativos, agentes colaborativos que aprenden, agentes de interfaces y verdaderos agentes inteligentes.

Enfatizamos en que estas distinciones no son definitivas. Por ejemplo, con agentes colaborativos hay mas énfasis en cooperación y autonomía que en aprendizaje; pero no implica que los agentes colaborativos nunca aprendan. Igualmente, para agentes de interfaz, hacemos más énfasis en autonomía y en aprendizaje que en cooperación. No hemos considerado las áreas que hay por fuera de las intersecciones entre agentes. Por ejemplo: la mayoría de los sistemas expertos y las bases de datos deductivas son muy autónomas, pero típicamente no cooperan, ni tampoco aprenden. Idealmente los agentes deberían tener igual cantidad de los tres atributos ideales, pero esto es más una aspiración que una realidad. Verdaderos agentes inteligentes no existen actualmente desarrollados.

También pueden ser clasificados por el rango de sensibilidad de sus sensores o por el rango y efectividad de sus acciones, o por la cantidad de estados internos que poseen y deben haber muchas más posibilidades.

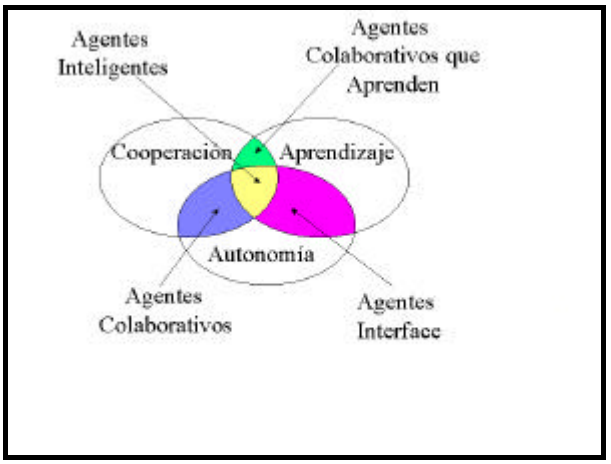

Figura 2.

6. Arquitectura interna.

El objetivo de la IA es el diseño de un programa de agente: una función que permita implantar un mapeo, para que el agente pueda pasar de percepciones a acciones. Dicho programa se ejecutará en algún tipo de dispositivo de cómputo, al que se denominará arquitectura. En general, la arquitectura se encargará de lo siguiente:

Ejecutar el programa agente, poner al alcance del programa las percepciones obtenidas mediante los sensores y alimentar los efectores con las acciones elegidas por el programa conforme éstas se van generando, bien sea de una base de datos deductiva o de un sistema experto con una base de conocimiento (Fig.3).

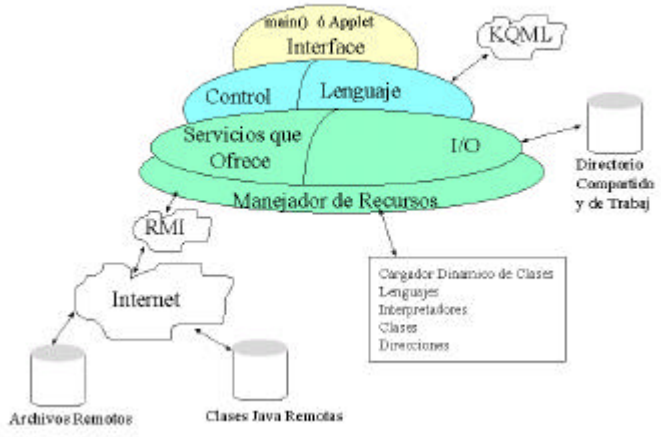

Figura 3.

La arquitectura debe ofrecer un cierto grado de aislamiento entre los dispositivos sensores y efectores que facilite la programación a alto nivel.

Con dicha arquitectura de agente, ahora exploremos como se podría lograr la comunicación entre diferentes agentes y de qué manera estos podrían colaborar.

7. Comunicación entre Agentes.

En general la comunicación es el intercambio intencional de información que se lleva a cabo mediante la emisión y percepción de signos pertenecientes a un sistema convencional de éstos.
Una de las acciones que puede realizar un agente es la de producir lenguaje.

¿Que podría motivar a un agente realizar actos de comunicación? Supongamos que un grupo de agentes está explorando el web. El grupo se beneficia a nivel colectivo e individual cuando es capaz de realizar alguna de las siguientes acciones:

- Intercambiar Información entre sí sobre aquella parte del web que cada uno de ellos ha explorado, lo que les ahorrará a todos trabajo de exploración. Para ello

se valdrán de *aseveraciones* como: Hay algo de la contaminación del río Magdalena aquí.

- Consultar a otros agentes sobre determinados aspectos del web. Generalmente se utilizan para ello *preguntas como*: ¿Has encontrado algo acerca de Marte?.
- *Responder preguntas*: Se trata de una forma especial de dar información. Sí, en la siguiente dirección encontraras la misión PATHFINDER.
- *Peticiones u órdenes formuladas a otros agentes para que realicen determinadas acciones*: Por favor, podrías enviarme una fotografía del Rover.
- *Promesa de hacer algo*: ¡Si lo haré!
- *Proponer tratos*: Lo haré si tu me envías una de Saturno.
- *Aceptación de peticiones y propuestas*: De acuerdo.
- *Compartir sentimientos y experiencias entre sí*: La misión PATHFINDER es fabulosa.

La comunicación y cooperación entre agentes descrito anteriormente se facilita si se usa un lenguaje común. En términos lingüísticos, esto significa que se comparta una sintaxis, semántica y pragmática comunes. Para ello es necesario dotar a los agentes de una capa encargada del manejo de dicho lenguaje común. Los agentes entonces pueden realizar acciones de comunicación, las cuales a su vez se convierten en percepciones para otro agente, con el cual se desee comunicar.

KQML Agent Comunication Language.

En IA el compartir una sintaxis común es un problema, debido a que no hay un lenguaje aceptado universalmente en el cual representar conocimiento, información y consultas. Lenguajes tales como KIF ó SQL extendido, son aceptados pero no se han convertido en estándares. Como resultado de esto, es necesario decir que dos agentes pueden comunicarse el uno con el otro, si tienen una representación común del lenguaje o usan lenguajes que se pueden traducir el uno al otro.

Asumiendo un lenguaje común o traducible, es necesario aún que compartan un frente de trabajo o conocimiento para interpretar los mensajes que intercambian. Esta no es realmente una semántica compartida, pero si es una ontología compartida, lo cual es mucho. Pragmática entre procesos de cómputo incluye: conocer con quien hablar y como encontrarlo y conocer como iniciar y mantener un intercambio

Como lenguaje de comunicación común, para la implementación de agentes actuales, se utiliza el lenguaje KQML (Knowledge Query and Manipulation Language). KQML es un lenguaje diseñado para soportar interacciones de agentes inteligentes. Es tanto un formato de mensajes como un protocolo para el manejo de tales mensajes, soportando el intercambio de conocimiento entre agentes. Está constituido por un pequeño número de directivas, las cuales son usadas por los agentes para describir los metadatos y especificar los requerimientos de información.

Ahora que tenemos un agente capaz de manejar mensajes, exploremos como se pueden comunicar los agentes, que se encuentran distribuidos en una red local, probablemente en un host de Internet.

Para ello es necesario introducir una clase especial de agente, llamado facilitador de la comunicación, el cual se encarga de coordinar la comunicación entre los demás

agentes, desarrollando varios servicios, pasar mensajes a los agentes, enrutar mensajes basándose en los contenidos, encontrar información para los clientes y ofrecer mediación y traducción de servicios. El agente facilitador puede hacer esto, ya que cada agente cuando se inicia, envía al agente facilitador, su nombre, su dirección IP y los servicios que puede ofrecer a los demás agentes en la red.

Como un ejemplo, (vease la figura 4) considere un caso donde un agente A desearía conocer si una proposición X es verdadera o no. Un agente B podría tener X en su base de conocimiento, y un agente facilitador F esta disponible.

Si A sabe que es apropiado enviar una consulta acerca de X a B, entonces puede usar un simple protocolo punto a punto y enviar una consulta directamente a B. Si por el contrario A no sabe qué agentes están disponibles, o cual de todos podría tener X en su base de conocimientos, o comó contactar a algún agente, entonces A le pide a F que le recomiende un agente que sea apropiado para enviarle una consulta acerca de la verdad de X. Entonces el facilitador F le responde a A con el nombre y la dirección del agente B. A queda entonces libre para iniciar un dialogo con el agente B, para enviar ésta u otras consultas similares.

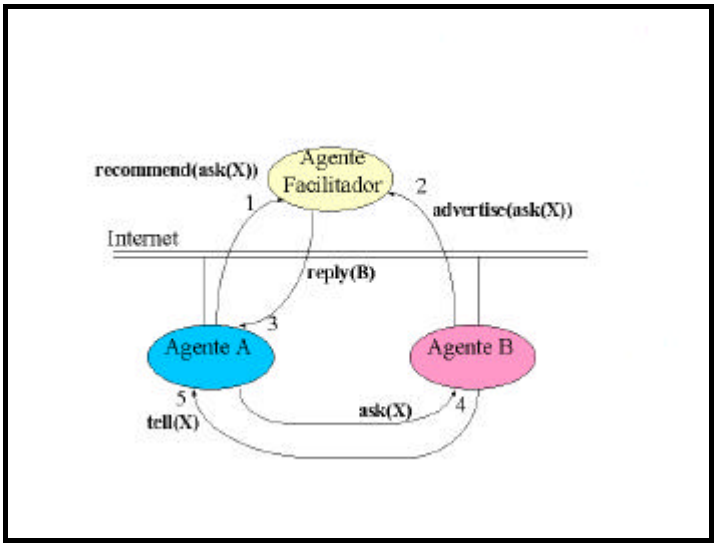

Figura 4.

Podemos observar que una de las principales funciones de un agente facilitador es la de ayudar a otros agentes a encontrar clientes y servidores apropiados. El problema de cómo los agentes encuentran facilitadores en primer lugar, no es un problema de KQML y hay una variedad de posibles soluciones.

Actualmente las aplicaciones basadas en KQML usan una técnica simple. En el proyecto PACT, por ejemplo, todos los agentes usan un facilitador central común, cuya localización fue un parámetro de inicialización cuando los agentes fueron lanzados.

En las aplicaciones ARPI, el actual Servidor de nombres de Dominio (DNS Domain name Servers), de Internet, es una muy simple y robusta técnica para mapear nombres simbólicos a direcciones IP de Internet. Técnicas similares pueden ser usadas

para mapear nombres simbólicos de agentes en referencias que pueden ser usadas para contactar un agente determinado.

8. Implementación.

Ahora desde una perspectiva práctica, estamos implementando totalmente en Java una propuesta de interfaz para proyectos colaborativos, utilizando características nuevas como RMI, JDBC y programación multihilos. Será una herramienta de software distribuida que servirá como apoyo a la interfaz gráfica "La PachaMama" por lo que el público objetivo son profesores y estudiantes tanto de primaria como de secundaria, interesados en proyectos de apoyo tecnológico. Asimismo se propone una intervención por parte del público externo permitiendo el acceso de diferentes tipos de personas que apoyen y complementen el proceso educativo, enfocado primordialmente a la construcción a partir del Alumno con el apoyo activo del Maestro.

Ambos, Maestros y estudiantes, son vitales para el desarrollo y los logros óptimos de esta interfaz, ambos deben participar ya que cada día se les brindan medios y métodos más eficaces que enriquecen el proceso de la Educación. Además es importante que este tipo de procesos se implementen dentro de las instituciones educativas.

La interfaz comienza con una entrada que define los Proyectos Colaborativos que posteriormente ubica al usuario dentro de un ambiente de trabajo motivador, donde encuentra un panel de control con herramientas basadas en el concepto de Trabajo Colaborativo. La interfaz puede observarse en la figura 5:

Figura 5.

Las herramientas se dividen en varios tipos, dependiendo del papel que cumple cada una dentro de la interfaz:

Herramientas de tipo administrativo:

Estas herramientas cumplen la función de controlar y ayudar al usuario dentro del proceso.

Agenda

Aquí se presentarán diferentes tipos de formatos para las actividades a realizar, concretar y definir dentro de cada proyecto que se tratará, fechas, tiempos y quién convoca las conferencias o encuentros y como se puede o debe trabajar.

Cronograma

El manejo de cronograma se podrá dar de manera individual o grupal, dependiendo de cómo se esté desarrollando la actividad, por medio de un formato que permitirá almacenar datos de la organización y el manejo de tiempos por parte del usuario(s), lo que le permitirá tanto al Alumno como al Maestro llevar un orden y seguimiento de todo el proceso.

Lista de actividades

Es una ayuda que le permitirá al usuario ver el orden de herramientas que ha utilizado durante la actividad, para navegar entre ellas sin perder la información en cada una y permitir que el trabajo no se base únicamente en el uso de una herramienta, pues la propuesta es aprender a integrarlas.

Seguridad

Aquí el Maestro o el Administrador del sistema encontrará un formato por medio del cual podrá definir los usuarios y los grupos que van a trabajar dentro de la actividad, clasificar a los usuarios dentro de los diferentes grupos y definir qué herramientas podrán utilizar.

Ayuda

Esta le explicará al usuario de manera gráfica y por medio de ejemplos el manejo de cada una de las herramientas.

Administrador

Es el espacio donde los Maestros se podrán encontrar y ver el trabajo realizado por los Alumnos, y complementar la información.

Herramientas de identificación

Aquí se ubica al usuario dentro del contexto de usuarios del mundo y se muestra él mismo hacia el mundo, estas herramientas son :

Tarjeta de identificación

Aquí se presentará un formulario con información concreta que deberá ser rellenado cuando comience la actividad ; los datos necesarios son el nombre del usuario, su dirección de correo electrónico, su clave para ingresar posteriormente a su página y el trabajo colaborativo que va a realizar. Esta herramienta se conectará inmediatamente con el mapa y a la vez con la seguridad, la cual determina las herramientas. Esta tarjeta permite que las demás personas puedan identificarlo y realizar una comunicación más amena, familiar y directa con él, etc.

Mapa

Éste identifica al usuario en el mundo, y nos muestra la ubicación de quienes están conectados. Al seleccionar un usuario, me da la información necesaria anteriormente descrita en la tarjeta de identificación, y el usuario puede tomar la decisión de conectarse o no con quien lo considere necesario.

Herramientas de Comunicación

Son estas las que permiten desarrollar ampliamente la tarea dentro del tablero compartido. Estas son :

Texto y Graficador

Como su nombre lo indica permiten hacer textos y gráficos. Si el usuario lo considera importante, en esta interfaz se podrá trabajar con otros usuarios simultáneamente.

Chat

Esta herramienta permite una comunicación por medio de textos en un cuarto dentro del tablero de manera compartida entre dos o mas usuarios.

Vídeo Conferencia

Aquí se permite enviar o trabajar sobre videos, pudiéndose ver en diferentes partes de manera simultánea.

Correo electrónico

Es una forma de comunicación Asincrónica que permite trabajar en diferentes tiempos, pero que potencializa y fomenta la investigación.

Herramientas para actividades

Estas herramientas permiten potenciar los procesos que se trabajan dentro de la institución y la familia, ellas llevan un formato flexible en el cual se puede trabajar de una manera dinámica y constructiva además de ser es donde se almacena toda la información pedagógica de contenidos para las actividades.

Diario de Procesos

Aquí es donde se encuentran inscritas las Unidades de Aprendizaje Integrado y las diferentes páginas de trabajos colaborativos propuestos desde el Proyecto Conexiones.

Actividades

Aquí se encuentran actividades propuestas como el Periódico electrónico desde donde se puede ver y a la vez actualizarlo, Carreras de observación dentro del web, etc.

Evaluación

En este espacio, tanto Maestros como Alumnos podrán realizar una actividad conjunta de evaluación tanto de ellos mismos, de la clase y de los proyectos que realizan.

Nuevo / Juegos

Es aquí donde los estudiantes y Maestros proponen nuevas alternativas y metodologías gracias a la inquietud generada dentro del proceso de aprendizaje. También encontrarán enlaces con proyectos colaborativos Internacionales.

Asimismo se proponen juegos Colaborativos que fomentan la comunicación y las relaciones que permiten un crecimiento en valores.

Tablero compartido

Es el punto fundamental de la propuesta ya que es aquí donde convergen todas las herramientas ya sean de uso libre o por medio de formatos. Es dónde el usuario interactúa y se ayuda a formar como ser social e integral. En este tablero iremos añadiendo nuevos componentes, conforme se vayan desarrollando, como por ejemplo: el graficador y el chat que ya se encuentran en etapa de evaluación (Fig 6). En él varios usuarios tienen la posibilidad de graficar simultáneamente sobre la misma área de trabajo e incluso conversar vía chat acerca del dibujo que están realizando. Cuando algún usuario en alguna institución asociada al proyecto Conexiones haga una raya, un circulo o un rectángulo, dicha información será capturada por un agente y enviada al servidor, donde el agente facilitador la recibirá y almacenará. Cada cierto tiempo los agentes en las máquinas clientes pedirán al agente facilitador que les envíe la información más actualizada de los dibujos y mensajes que han realizado los demás usuarios, como puede observarse en la figura 6, el resultado es que todos los usuarios comparten la misma vista y pueden trabajar simultaneamente en busca de lograr los objetivos del proyecto colaborativo.

Como hemos mencionado ya han sido implementados agentes de software, que se comunican a través de Internet, facilitando a su vez la comunicación entre los diferentes usuarios, que usan dicho tablero Es así como se tiene una infraestructura básica en la cual los agentes se registran con un agente facilitator situado en el servidor del Proyecto Conexiones usando un nombre y una clave, y a través de éste se conectan y desconectan de Internet, envían y reciben mensajes, transfieren archivos, e invocan otros programas que están corriendo en otros computadores. Ha sido desarrollado totalmente en Java para que pueda ejecutarse en plataformas heterogéneas y hacer uso de applets livianos como agentes temporales, solucionando así el problema de espacio en las instituciones educativas.

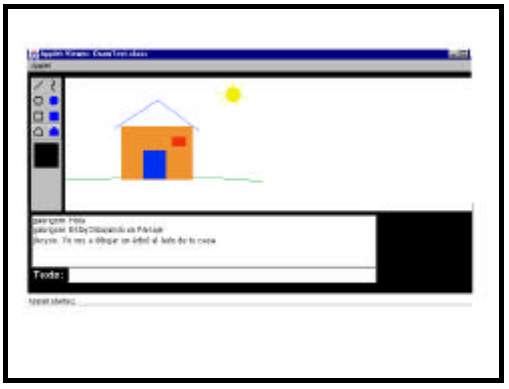

Figura 6.

Todos los agentes envían y reciben mensajes vía el agente facilitador, el cual los redirecciona a sus destinatarios. Cuando un agente intencionalmente se desconecta, o accidentalmente se bloquea, el facilitador continúa almacenando los mensajes que

llegan hasta que el agente se reconecta. Es decir, almacena todos los mensajes, tal como un servidor de email. Así los mensajes no se pierden debido a problemas temporales de la red. Esto también permite a los agentes individuales desconectarse de la red y retornar por sus mensajes en un tiempo futuro. El facilitador es particularmente importante para agentes applets, los cuales pueden únicamente iniciar conexiones vía socket con el host del cual se bajó, debido a las restricciones de seguridad de Java.

Hay un resultado muy importante que se obtiene al utilizar RMI y es el hecho de que el agente facilitador permitirá a los agentes de Java intercambiar mensajes con cualquier otro agente registrado en Internet, haciendo una "única conexión" con el agente facilitador, el cual redirecciona los mensajes sin que el emisor tenga o conozca la dirección del agente receptor y sin hacer una conexión socket separada, como es usual en las implementaciones actualmente conocidas.

Los agentes todavía no cuentan con capacidades específicas. Sólo las necesarias para comunicarse e interactuar, sin embargo, ya que se ha utilizado una nueva característica de Java, llamada RMI, creemos tener una base robusta para investigar y construir tales agentes inteligentes. En particular, se están desarrollando "agentes inteligentes" que busquen información. Dichos agentes de información tendrán potencialmente acceso a muchas fuentes de información y serán capaces de confrontar y manipular información de esas fuentes para responder a consultas hechas por el usuario o por otros agentes de información inteligentes. Las fuentes de información pueden ser de muchos tipos, por ejemplo, bases de datos tradicionales. El agente podrá ser enviado a investigar, y, después de una cuidadosa búsqueda en varios sitios FTP, retorna con un informe técnico apropiado, con los detalles de los contactos involucrados en la investigación. Además se está investigando la manera de proveer la infraestructura que permita a los agentes moverse de una máquina a otra.

JDBC permite que desde aplicaciones o Applets de Java, se puedan establecer conexiones a Bases de Datos, en combinación con RMI (Fig 7). Podríamos tener una base de datos en el servidor del proyecto Conexiones, a la cual desde cualquier institución, los niños podrían ejecutar consultas y actualizaciones, sin necesidad de tener en dichas instituciones ningún manejador de Bases de Datos instalado. La implementación realizada hasta el momento, consiste de una sencilla base de datos que almacena los nombres, identificación y claves de los usuarios autorizados para utilizar la interfaz de proyectos colaborativos. Dicha base de datos reside en el servidor del Proyecto, entonces al iniciar cada aplicación cliente, ésta le solicita al usuario la identificación y la clave tras conectarse al servidor de Conexiones, le envía dichos valores al servidor para que sea éste quien verifique que el usuario existe y que la clave es la correcta.

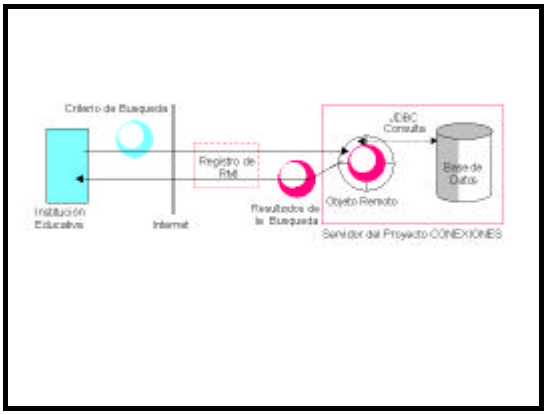

Figura 7.

RMI (Fig 8) provee una solución elegante para que los objetos residentes en diferentes máquinas se comuniquen, brindando las facilidades necesarias para un amplio rango de aplicaciones distribuidas. Su principal objetivo es hacer que el proceso de invocación de métodos de objetos remotos sea tan similar como sea posible a la invocación de métodos de objetos locales, permitiendo a los objetos en una maquina virtual Java llamar a métodos que residen en otra maquina virtual, realizando muy pocos cambios en el código local y en el remoto. La principal diferencia desde la perspectiva del usuario es la necesidad de manejar las excepciones adicionales que pueden ser generadas por un objeto remoto, principalmente las relacionadas con la comunicación.

RMI puede ser visualizado como una arquitectura de tres capas, donde cada capa provee servicios a las capas anteriores.

Cuando un cliente invoca un método remoto, la llamada fluye hacia abajo a través de las capas, siendo transformada a medida que esto sucede, hasta crear un flujo de datos que fluye a través del transporte, hasta llegar al servidor. Allí es entonces reconstruido mientras sube a través de las capas, emergiendo como una llamada a un método local. Los resultados realizan el proceso inverso y finalmente el control es devuelto al código en el cliente que realizó la llamada.

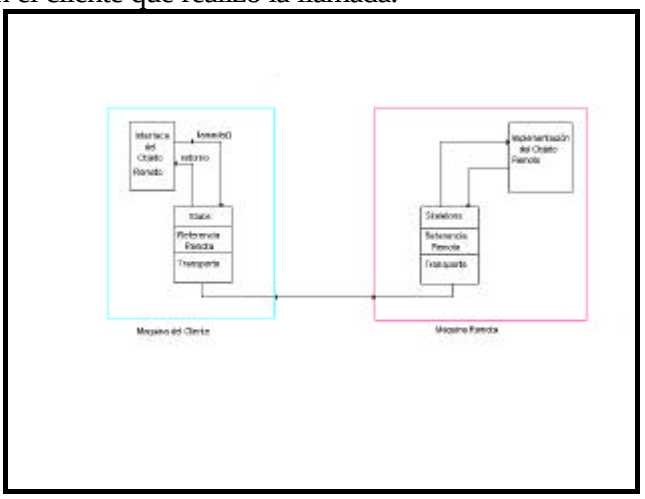

Figura 8.

RMI emplea un buen número de mecanismos y técnicas muy interesantes para implementar Objetos Distribuidos, una de ellas bien vale la pena mencionarla y es conocida como Carga Dinámica de clases: RMI toma ventaja del Bytecode de Java para bajar el código que necesite desde el servidor, pudiendo inclusive obtener el código completo del cliente; esto se logra mediante el RMIClassLoader y un manejador de seguridad RMISecurityManager, para asegurar que las clases se bajan únicamente, si no están disponibles localmente, y que son obtenidas de una fuente confiable. Esta técnica nos permitirá en un futuro, que si un niño dibuja una esfera, y en su máquina no existe ningún código capaz de hacer rotación de objetos, podrá encontrar y obtener dicho código, desde cualquier nodo de la red que este participando en el proyecto, bajarlo, instanciarlo y ejecutarlo, todo esto de una manera transparente para el usuario.

9. Conclusiones.

En el pasado, la computación distribuida era de interés solo para un pequeño número de programadores, muchos de los cuales trabajaban para grandes compañías con

grandes redes. Con la disminución del costo de las conexiones a Internet, los computadores personales tienen acceso a grandes recursos. Es así como los recursos tecnológicos en la escuela han cambiado y evolucionado, lo que necesariamente debería llevar a los cambios en la forma de estructurar el aprendizaje de los alumnos. Estamos conscientes que en nuestros alumnos es necesario fomentar la creatividad, la responsabilidad individual, el trabajo colaborativo, la capacidad crítica, etc.

La metodología por proyectos es una estrategia de aprendizaje que permite la estructuración del aprendizaje en torno a actividades relevantes para los alumnos y que engloben diferentes tipos de objetivos y contenidos, estimulando aspectos cognitivos, motrices, éticos y afectivos. Las nuevas tendencias tecnológicas como Java con características tales como RMI y JDBC, son un catalizador para que se desarrollen en poco tiempo y a bajo costo una nueva generación de software distribuido que apoye los objetivos del trabajo colaborativo acortando tanto las distancias geográficas como tecnológicas al permitir la vivencia y la interacción entre alumnos de diferentes regiones o países.

10. Requerimientos.

La interfaz para proyectos colaborativos puede correr en cualquier plataforma que soporte el Java development kit JDK1.1 de Sun Microsystems Inc., incluyendo Windows95, WindowsNT, Solaris, Mac OS8. Visualizadores tales como Netscape , Internet Explorer, o el applet viewer de SUN pueden ser utilizados. Se asume que todas las comunicaciones son a través de TCP/IP.

11. Bibliografía.

- Software Agents, Editado por Jeffrey Bradshaw, Publicado por AAI Press/The MIT Press, 500 pp.
- BRUSTOLONI, José C. Autonomous agents: characterization and requirements. Noviembre 1991. School of Computer Science. Carnigie Mellon University. Pittsburgh, PA
- Downing, Bryan . RMI: Developing Distribute Java Applications With Remote Method Invocation and Object Serialization.
- Franklin, Stan (1997). Artificial Minds, MIT PRESS.
- Cassady-Dorion, Luke (1997). Industrial Strength Java. Capitulo 12, Distributed Computing with Java.
- Galvis, Alvaro. Ingeniería de Software Educativo. Universidad de los Andes.

http://www.cs.umbc.edu.agents
http://ai.eecs.umich.edu/cogarch0/