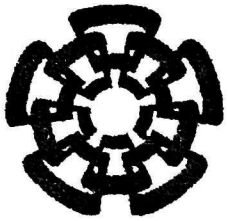

CINVESTAV

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara de Ingeniería Avanzada

AMBIENTE GENÉRICO VIRTUAL DISTRIBUIDO GeDA-3D

Tesis que presenta
Lic. Silvia Inés Toscano Garibay

Para obtener el grado de
Maestro en Ciencias

En la especialidad de
Ingeniería Eléctrica

Guadalajara, Jal., octubre de 2000

Bibliograficos

AMBIENTE GENÉRICO VIRTUAL DISTRIBUIDO GeDA-3D

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:

Silvia Inés Toscano Garibay

Licenciada en Informática
Universidad de Guadalajara, 1994 - 1998

Becario del CONACYT, expediente no. 129222

Director de tesis

Dr. Félix Francisco Ramos Corchado

CINVESTAV del IPN Unidad Guadalajara, octubre de 2000

Ambiente Genérico Virtual Distribuido

Lic. Silvia Inés Toscano Garibay

19 de septiembre de 2000

Índice General

Capítulo 1

Introducción

1.1 Objetivo

En este capítulo se describe el problema que nos interesa resolver con este trabajo de tesis. Posteriormente, se expone la solución propuesta y sus características mas importantes, identificando así la magnitud del resultado esperado. Para terminar se describe la organización de este documento.

1.2 Descripción del problema

El problema que nos ocupa en este trabajo es la dificultad de desarrollar sistemas distribuidos cooperativos, es decir sistemas computacionales que ayuden a los seres humanos a interactuar con otros seres humanos o con elementos del sistema que ayuden a realizar un trabajo.

La sociedad ha experimentado cambios significativos en cuanto a su organización y el papel de la computación se hace evidente en estos cambios. Por ejemplo, en la toma de decisiones de una organización, no es suficiente poder establecer una comunicación entre personas, sino que se hace necesaria la comunicación de información, en grandes y pequeños volúmenes de manera eficiente y segura entre uno o varios elementos del sistema. Este pequeño

ejemplo muestra la inmersión de la informática en la vida del ser humano en nuestros días, y el cambio del papel que están jugando las computadoras, las cuales se utilizan mucho más para finalidades de comunicación que de cálculo o procesamiento como ocurría anteriormente.

El estado del arte en la informática ha alcanzado un punto en el que existen diferentes metodologías que permiten el desarrollo de sistemas distribuidos aplicados a muchas actividades, tales como entretenimiento, control de información, CSCW[1], inteligencia artificial, comercio, etc. En general estas metodologías nos han conducido a desarrollar módulos, llamados de diferentes maneras como son los procedimientos o funciones[53], objetos[3], o recientemente agentes[4][5]. Sin embargo, siguen existiendo muchos problemas en la obtención de un sistema funcional a partir de estos módulos. Existen diferentes plataformas y lenguajes como son: CORBA[6][65][15][18][19][20], DCOM[21], Java[22][23][24][25] y TCL[26], los cuales ayudan a resolver problemas de interoperabilidad, diferencias de lenguajes de implantación o entre las plataformas de desarrollo, etc. Es decir, estas herramientas ayudan a resolver los conflictos que surgen en el proceso de comunicación. Pero el problema real es cómo facilitar la cooperación, ya que ésta significa control y coordinación, las cuáles son indispensables en cualquier desarrollo de sistemas.

1.3 Solución

La principal motivación de nuestro trabajo, es proponer una herramienta que nosotros llamamos GeDA-3D, la cual tiene como característica principal, facilitar el desarrollo de sistemas, donde la cooperación es importante en el logro del objetivo. Es decir, nosotros buscamos trascender con respecto al software existente, por ejemplo, aquellos sistemas basados en CORBA y DCOM[27] y que fungen como medio de comunicación entre las personas. Estos sistemas ofrecen también herramientas para el alcance de un objetivo en

conjunto[1] De lo anterior, podemos asegurar que nuestro trabajo es diferente al del desarrollo de una aplicación con características similares a las de los mencionados, en el sentido de que tratamos de concebir una única plataforma que constituya la base de aplicaciones distribuidas cooperativas versátiles y con la posibilidad de utilizar una interfaz que puede ser incluso un ambiente diseñado en 3D.

Para lograr proponer una arquitectura genérica, que es el objetivo de esta tesis se realizó un estudio comparativo de las arquitecturas de algunos sistemas distribuidos que trabajan con TCP/IP[28], esto es, en redes de computadoras homogéneas o heterogéneas, como en internet; se analizaron algunos trabajos sobre el trabajo cooperativo, se realizó investigación de técnicas de cooperación y se realizaron juntas de trabajo con el objetivo de obtener una lista de los requerimientos de nuestra arquitectura. Bajo este esquema, un programa distribuido, debe poder adaptarse a esta plataforma la cual le proporcionará las habilidades de comunicación e integración entre sus propios componentes.

Entre las características de la plataforma genérica propuesta están:

1. La arquitectura debe facilitar la implementación de cualquier aplicación distribuida o centralizada que necesite interacción entre sus componentes;

2. Interoperabilidad, para asegurar que la funcionalidad del sistema sea correcta en ambientes multiplataforma; es decir, funcional independientemente del hardware y sistemas operativos que posee el equipo de cómputo;

3. Facilidad en la administración de las relaciones de cooperación entre los elementos del sistema, es decir, con los mecanismos que facilitan la administración de todos sus elementos;

[1]Estas herramientas, utilizadas por sistemas afines, serán enumeradas en el capítulo 2.

4. Ya que el aspecto de la interfaz es muy importante para la aceptación de un sistema, la arquitectura da facilidades de creación de una interfaz tridimensional, con el fin de presentar una apariencia amigable y confiable que capture la atención del usuario.

5. La arquitectura representa una base sólida e independiente de la especialización de una aplicación. Constituye una plataforma integrada por un conjunto de componentes y funciones estándar para que, con una aplicación particular, pueda ser fácilmente adaptada a las necesidades de uno o varios usuarios.

6. Dada la diversidad de esta colección de aplicaciones, debemos aclarar que no es accesorio que sean amigables, debido a que se requiere que estén al alcance de la también diversificada población de usuarios. Para esto, hemos también integrado los mecanismos de comunicación dentro de un espacio tridimensional, que será el espacio de trabajo para el usuario. Éste, puede tomar distintas facetas de acuerdo a la naturaleza del sistema al que se aplique. El espacio de trabajo será un escenario con la capacidad de cambiar según las necesidades del producto, que como hemos mencionado con anterioridad, puede ser desde una herramienta basada en CSCW hasta un sistema de software de apoyo a la enseñanza.

Para validar la aplicabilidad de nuestra Arquitectura Genérica, se decidió implantar una Oficina Virtual, que es uno de los problemas más estudiados en sistemas de comunicación y cooperación. En una Oficina Virtual, se puede establecer completamente una organización empresarial de manera distribuida y obtener los servicios típicos de una oficina real. La selección de este sistema se debe a la experiencia del grupo en el desarrollo de este tipo de sistemas y también a que los resultados de este sistema en cuanto a restricciones temporales estrictas nos dan una idea de la eficiencia del sistema en casos menos restrictivos.

1.4 Organización de la tesis

La presentación de la información en este documento, respetará el siguiente orden:

1. El capítulo segundo, titulado Estado del Arte, expone los tópicos que conforman las bases teóricas del proyecto, tales como: los conceptos generales del funcionamiento de los sistemas distribuidos; el análisis de las organizaciones basadas en trabajo cooperativo; algunos aspectos relacionados con el aspecto gráfico del sistema y algunos atributos ergonómicos que facilitan en función de una adecuada percepción, el adecuado manejo de la aplicación, ya sea en la parte común de control y configuración de datos o en el espacio tridimensional de interacción con el usuario; finalmente, una revisión de sistemas existentes semejantes y su arquitectura en general, las ventajas y desventajas que se encontraron en ellos indicando cuales han influido en las decisiones que tomamos durante la implementación del caso de prueba de nuestro proyecto.

2. El capítulo tercero muestra específicamente los fundamentos que conforman la base de nuestra arquitectura, estas características nos proporcionan un claro panorama sobre los alcances del concepto.

3. El capítulo 4, muestra el uso, la flexibilidad, adaptabilidad de la arquitectura a través de un ejemplo. El caso escogido es una herramienta CSCW. Además, mostraremos cuales herramientas fueron seleccionadas para la implementación de ésta.

4. Enseguida, el capítulo 5 hace explicitas las conclusiones de la investigación realizada y aquellos proyectos que prometen ser productivas extensiones de nuestro trabajo, hacia el futuro.

5. Finalmente podremos encontrar, un apéndice con información útil para utilizar nuestro modelo genérico, en términos prácticos y con breves

descripciones del código que un usuario debería realizar a fin de que su implementación sea satisfactoria. Se muestra también en el apéndice, algunos diagramas de la metodología seguida(UML[29]).

Capítulo 2

Fundamentos

2.1 Objetivo

El objetivo de este capítulo, es exponer un resumen de los tópicos teóricos que se convirtieron en los fundamentos para el progreso de nuestra investigación, tal como el paradigma de los sistemas distribuidos[30], una introducción al trabajo cooperativo soportado por computadora (CSCW)[31], los conceptos básicos de Realidad Virtual[23][24][27][32] así como una revisión de los trabajos similares a nuestro caso de prueba.

2.2 Introducción

El desarrollo de nuestro estudio, involucra diferentes aspectos del área de la informática, en este capítulo describimos los conceptos básicos de algunas tecnologías que han servido para alcanzar nuestro propósito. Comenzaremos con los conceptos de sistemas distribuidos, de los cuáles es necesaria su descripción ya que esta es la naturaleza de los proyectos que pueden ser respaldados por el tema de tesis que nos ocupa. En seguida, dado que en la última década, la tendencia natural del software se centra sobre un esquema cooperativo, incluimos la definición y alcances de CSCW (Computer

Supported Cooperative Work) porque forma parte esencial de las bases del presente trabajo. Finalmente, calificaremos nuestro proyecto en función de los criterios proporcionados por la Realidad Virtual, puesto que dentro de las características de nuestra propuesta esta la facilidad de utilizar una interfaz tridimensional, en particular, se trata de un escenario tridimensional compartido basado en el tiempo[1].

2.3 Conceptos

2.3.1 Sistemas Distribuidos

Características de los Sistemas Distribuidos

Existen diferentes definiciones de sistema distribuido (SD), en general se hace referencia a la interconexión de un grupo de computadoras que interactúan a fin de alcanzar un mismo objetivo[30].

De las definiciones consultadas por el grupo, concluimos que las tres principales características de los SD que nos interesan son:

1. *Múltiples computadoras*: En general, debe ser más de una computadora, cada una con al menos una UCP, memoria local, tal vez un medio estable de almacenamiento como discos y medios de entrada y salida de información para conectarse al ambiente.

2. I*nterconectividad*: Los medios de comunicación de entrada y salida deben permitir la conexión entre cada terminal y el resto de las computadoras del sistema.

3. *Un estado global compartido*: Debido a que este conjunto de computadoras coopera para mantener un único estado del sistema.

[1] Debemos aclarar que basado en el tiempo no tiene el mismo significado que tiempo real. La relatividad de las actividades con respecto al tiempo es mas estricta en el segundo caso. Solamente suponemos que debe existir una linea de tiempo que sea la que proporcione un sentido común a las entidades participantes del ambiente distribuido.

Además, existen factores característicos como la independencia de fallas, comunicación no confiable, insegura o costosa que deben observarse para que un SD alcance las expectativas de un desarrollador.

El éxito de los sistemas distribuidos actualmente reside en la naturaleza de las aplicaciones, las cuales necesitan compartir información y otros recursos geográfica y organizacionalmente distribuidos. Un ejemplo de un SD de propósito general es un sistema de red, donde colaboran un conjunto de terminales y servidores interconectados formando una sola estructura capaz de conectarse con otras estructuras o redes.

Otro factor que debe considerarse dentro del desarrollo de un SD es la seguridad. La seguridad es un factor muy difícil de controlar, independientemente de si el sistema es centralizado o distribuido. En un sistema centralizado, la vulnerabilidad o inseguridad es una condición presente, porque cualquier acción que logre violar la única barrera que representa su núcleo de operación, por ejemplo el sistema operativo, permite un acceso relativamente fácil y abierto al sistema completo. En cambio, aunque un sistema distribuido es también propenso a fallas de seguridad, debido a que puede ser violado a través de cada uno de sus componentes, el alcance del perjuicio está limitado por el alcance en comunicación o control definido en cada componente.

Modelos en los Sistemas Distribuidos

Para diseñar correctamente un sistema, de cualquier tipo, hace falta algo más que la intuición. Sabemos que los modelos proporcionan la información necesaria para respetar un formato de comportamiento y / o estructura en determinadas circunstancias. Digamos que un modelo es una versión limitada del patrón real que incluye un conjunto de propiedades o atributos y una colección de reglas que definen un comportamiento, es decir, la forma en que se usarán los atributos al interactuar con el exterior.

La implantación del modelo de comportamiento o protocolo influye en

la parte mas crítica de la implementación de un sistema distribuido que es el diseño eficiente de sus capacidades de coordinación. Estas capacidades pueden ser la identificación del número y rol de las entidades coordinadoras pasivas o activas y la determinación del número de niveles jerárquicos de coordinación.

Otro factor que influye en la definición del esquema de coordinación y por lo tanto en el modelo es saber si el sistema será síncrono o asíncrono o incluso una mezcla de estos comportamientos.

En un sistema distribuido asíncrono no hay dependencia de información relativa a la velocidad de ejecución de un proceso y / o los retardos existentes en el envío de un mensaje. En cambio, cuando se trata con un sistema distribuido síncrono estos parámetros son muy importantes. En particular, la velocidad relativa a los procesos, así como cualquier retardo sufrido a causa de los canales de comunicación. En teoría, respetando lo anterior, podemos afirmar que todos los sistemas distribuidos son asíncronos a menos que utilicen o implementen planes u horarios que deban respetar durante su ejecución.

Otro factor que debe ser tomado en cuenta en el modelado de un SD son los tipos de fallas que deben tomarse en cuenta. En este aspecto, el comportamiento de falla no reside sobre el sistema completo, sino que se identifican aquellos componentes que de alguna forma ocasionaron la falla. A este respecto se dice que un sistema es t-tolerante a fallas, si el sistema satisface su especificación aunque un número máximo de t componentes fallen.

Definir los criterios para clasificar este tipo de fallas es delicado, porque pueden existir ambig üedades que impidan corregirlas. Por ejemplo, si un mensaje se pierde esto puede ser ocasionado por el componente que lo envió, el que lo recibió y/o por el canal de comunicación. Este ejemplo, nos muestra como una ambig"uedad nos puede llevar a tener conclusiones erróneas sobre el funcionamiento de un sistema t-tolerante a fallas. Para solucionar este

problema, se define el modelado de fallas en base al comportamiento de los componentes. A continuación se presentan los modelos de fallas[30] definidos:

1. Cuando el procesador se detiene completamente y permanece en ese estado se le llama Failstop. Esta falla es detectable por otros procesadores que participan en el sistema.

2. En la falla conocida como Crash el procesador también se detiene pero esto no es detectable por otros procesadores.

3. Crash + Link. Así se califica una falla en la que el procesador se detiene pero existe un canal de comunicación que presenta pérdida de mensajes.

4. Cuando un procesador recibe sólo un subconjunto de los mensajes que se le enviaron tiende también a detenerse, y esta falla se clasifica como omisión en la recepción.

5. La omisión en el envío se presenta cuando el procesador se detiene porque se ha enviado solamente un subconjunto de los mensajes programados.

6. Cuando se presentan fallas por omisión de envío o recepción y se detiene el procesador, entonces se califica como omisión general.

7. Finalmente las fallas bizantinas, que muestran un comportamiento arbitrario.

En algunos procesadores es poco probable que se presenten fallas y cuando sucede, éstas son detectables. En otros, aún cuando existan se tiene un comportamiento correcto.

El diseño de un sistema distribuido de lo explicado previamente, considera aspectos de bajo nivel, principalmente de comunicación. La siguiente sección se ocupa del trabajo cooperativo. El desarrollo en el trabajo cooperativo resuelve problemas a un nivel más alto, es decir se basa en toda la serie de

modelos y de algoritmos necesarios para soportar una comunicación confiable para proponer soluciones que necesitan la cooperación de más de un agente.

2.3.2 Trabajo Cooperativo

En un SD donde un conjunto de individuos o agentes persiguen un objetivo en común, es necesario encontrar una forma en que las actividades de comunicación, cooperación, coordinación, competencia, negociación y solución de problemas sean menos costosas, es decir, que se realicen con el menor tiempo, la menor acción y si es posible con menor complejidad.

El trabajo cooperativo o CSCW presume de cumplir con estas características y por tanto es el operador en los SD que se encarga de estas actividades, a través de redes de computadoras, correo electrónico, newsgroups, videoteléfonos o charlas en línea (chat). La mayoría de las aplicaciones en CSCW están basadas en tareas de comunicación y cooperación.

Existen algunas clasificaciones para el CSCW:

1. Aquel en el que el grupo de personas trabajan a la vez, esto es, CSCW en tiempo real o síncrono; y por otro lado, en donde el grupo trabaja en diferentes horarios, o sea CSCW asíncrono.

2. Aquel en el que el equipo de trabajo reside en un mismo lugar (CSCW localizado) o cuando los participantes están en localidades geográficamente distribuidas (CSCW no localizado).

Podemos definir al CSCW como el campo de estudio que evalúa el diseño, adopción y uso del trabajo en grupo, cuidando especialmente los aspectos de competitividad, socialización e interacción. El CSCW se relaciona con algunas ramas de la ciencia de la computación, tales como conceptos de la interacción entre el humano y la computadora (HCI, Human Computer

Interaction[39]), redes, multimedios, conceptos de programación orientada a objetos, inteligencia artificial y ultimamente con la realidad virtual

Actualmente, existe una tendencia en el uso de las computadoras inclinada a labores de comunicación. El correo electrónico e internet constituyen la chispa para una gran explosión en el uso de la computadora.

Johansen[41] en 1984 afirmó que los sistemas computacionales serían principalmente usados para la comunicación y no para el cálculo. Hoy comenzamos a sospechar que Johansen tenía razón. Desde nuestro punto de observación personal, los antecedentes culturales y tradicionales de las personas están cambiando en cuanto al uso de las computadoras. Día a día, la llamada supercarretera de la información aumenta el volumen y complejidad de su estructura y esto definirá poco a poco el uso de las computadoras, tal vez en mayor forma que cualquier otro factor.

Objetivos de CSCW

Las múltiples metas del CSCW crean una definición mas precisa de esta materia, pero la siguiente es la razón que consideramos mas importante al estudiar CSCW:

El objetivo del trabajo cooperativo es descubrir distintos usos para la tecnología computacional con el objetivo de mejorar el proceso de trabajo en grupo a través del tiempo o del espacio.

Alcanzar esta meta en un sistema, requiere de la aplicación de ciertas nociones, tales como:

WYSIWIS (What You See Is What I See). Que corresponde al fenómeno de que dos individuos con independencia de localización geográfica vean lo mismo en el mismo tiempo. La tecnología computacional permite a las pesonas interactuar y comunicarse en un ambiente de este tipo.

Administración del tiempo. Un producto de groupware que proporciona

los servicios de calendario diario de actividades y horarios de proyectos en grupo ayuda a todos los participantes en el manejo del tiempo dedicado.

Multimedios. CSCW toma ventaja de la capacidad actual para el manejo de gráficos y sonidos en un sistema computacional, porque esto proporciona interfaces mas naturales para colaborar con otros.

Programa de usuario final. Una herramienta de groupware debe ser suficientemente flexible y amigable para cambiar de acuerdo a las necesidades del usuario o de acuerdo a el tipo de trabajo colaborativo.

Realidad Virtual. Esta característica es adicional en un sistema de trabajo cooperativo y permite a las personas geográficamente distribuidas poder sentir si es necesario su estancia física en un mismo lugar, independientemente de su localización geográfica.

El diseño de herramientas de groupware involucra gran investigación, en cuanto a la forma en que un grupo de personas entiende los estímulos externos y se comporta al buscar un sólo objetivo. Este conocimiento adquirido, genera a su vez la necesidad de investigación que ayude a satisfacer las necesidades detectadas. Principalmente dos aspectos son importantes dentro de este marco: el primero es el trabajo de como asegurar la comunicación y aquellos aspectos que influyen en el sentido que interpreta el usuario de esta experiencia, por ejemplo, los retrasos y fallas de sincronizació durante el diseño. Es necesario aclarar que el trabajo cooperativo debe de verse como un proceso con vida, mas que como un diseño estático. Esto es, se deben tomar en consideración varios factores, por ejemplo: un grupo de millones de personas no se comportan de la misma forma que un grupo compuesto por decenas, lo que significa que el programa debería poder satisfacer a varios distintos tipos de grupos. Además, debe ser fácil de usar, mas que los sistemas

monousuario, debido a la distinta velocidad con que normalmente se usa una aplicación en comparación con la velocidad de una conversación.

En general, las personas desearían utilizar un sistema de CSCW por diversas razones:

1. Para obtener una comunicación mas rápida, clara e intuitiva;
2. Para tener comunicación desde donde no se tiene;
3. Para habilitar algunas actividades de telecomunicación;
4. Para ahorrar los costos de viajes;
5. Para reunir las experiencias y perspectivas de muchas personas;
6. Para formar grupos con intereses comunes
7. Para ahorrar tiempo y costos en la coordinación de un grupo de trabajo;
8. Para facilitar la resolución de problemas en un grupo;

Algunas aplicaciones para un sistema de groupware y trabajo cooperativo:

1. La administración y calendarización de asambleas de videoconferencia.
2. Juegos multiusuario que utilizan video en vivo y el software de charla para comunicarse;
3. Discusiones en línea sobre temas específicos llamadas, sesiones de newsgroups.

Parecería que aún estamos lejos de crear un sistema de trabajo cooperativo que integre distintos servicios de comunicación, pero las posibilidades evolucionan constantemente con cambios no solamente en el ámbito de la tecnología sino en la forma en que socialmente se desenvuelve el ser humano en sociedad.

Modelo Asíncrono

Como ya especificamos, un modelo de CSCW asíncrono es útil cuando el trabajo que realiza el grupo no requiere que los agentes involucrados trabajen al mismo tiempo. Un claro ejemplo de este tipo de trabajo es el correo electrónico. Esta tecnología consiste básicamente en el envío simple de mensajes entre dos personas, aunque a veces incluyen funciones adicionales como reenvío de mensajes, creación de listas de correo o mailing lists, ordenamiento automático y procesamiento de mensajes.

Las listas de correo y los newsgroups pareciera que tienen el mismo comportamiento que el correo electrónico. Sin embargo, difieren porque el correo trata del envío de mensajes entre grandes grupos de personas. En la práctica, la diferencia principal entre los newsgroups y las listas de correo es que en el primer caso sólo aparecen los mensajes de un usuario cuando han sido explícitamente solicitados, mientras que en las listas de correo los mensajes enviados están disponibles desde el momento en que el autor ha podido publicarlos.

Los sistemas de flujo de trabajo (WorkFlow) permiten un gran movimiento y ruteo de documentos, además pretenden facilitar el desarrollo de formas y el soporte para distintos roles y privilegios para los participantes.

El hipertexto en internet es un sistema de ligado de documentos de texto, que se convierte en una actividad cooperativa cuando un documento publicado evoluciona y cambia, a través de los días, como respuesta a quienes lo consultan. Se puede obtener información de quienes consultan la información, de qué clase es ésta y / o saber que tan frecuentemente es solicitada. Esta forma de acceso al conocimiento, da información que puede interpretarse como cierta consciencia de las personas que participan. Por otro lado, todos los usuarios tienen la capacidad de crear documentos de hipertexto lo que garantiza que tengan las mismas capacidades de interacción.

Los calendarios de grupo ayudan a la planificación, la administración de un proyecto y a la coordinación entre varias personas; ayudan también a

localizar a las personas en cualquier momento e identificar si se encuentran atendiendo asuntos privados. De tal forma, que hay más elementos para la programación de sesiones.

Modelo Síncrono

De la misma manera, que para el modelo asíncrono, un modelo síncrono o tiempo real es útil cuando es necesario que el grupo de personas trabajan a la vez. Los pizarrones compartidos permiten a dos o mas personas observar o dibujar en una superficie aún cuando no estén en el mismo lugar. Esto puede ser usado, por ejemplo, durante una llamada telefónica, donde cada persona hace sus propias anotaciones o bien, en conjunto se puede abordar un problema visual. Estos sistemas están diseñados para conversaciones informales, pero puede atender también tareas de diseño gráfico colaborativo, de publicación de documentos o en aplicaciones de ingeniería.

Los sistemas de comunicación basados en video cuentan con tipos de transferencia punto a punto en dos sentidos o en multipunto, esencialmente se trata de un sistema que depende de una linea telefónica y un medio visual. Sus aplicaciones son variables y por otro lado, tiene distinta importancia dependiendo de la actividad en la que se usan; es más útil al discutir información visual que cuando se usa en un videoteléfono.

Los programas de charla o "chats" permiten a muchas personas comunicarse con mensajes de texto en un espacio público. Las personas utilizan listas de foros o salas para encontrarse, éstos se identifican por un nombre, por su localización, por el número de personas que lo visitan y un tema de discusión. Algunos foros pueden restringir el acceso a determinados usuarios y dar privilegios a una persona como moderador de las discusiones.

Estos sistemas, se han convertido en una actividad interesante para muchas personas porque, finalmente, se obtiene en el texto toda la conversación y así cualquiera puede entrar al juego en cualquier momento. Se aplican también como apoyo al proceso de toma de decisiones en el desarrollo

de un proyecto, facilitan la lluvia de ideas, la crítica constructiva, votaciones y en la ponderación de eventos y alternativas.

Por otro lado, los juegos multiusuario son ya muy comunes y tienen una gran historia. Constituyen el ejemplo prototípico de muchas situaciones multiusuario que no son estrictamente cooperativas. Actualmente, se usan en conjunto con la tecnología de los multimedios (chat, audio y video).

2.3.3 Realidad Virtual

La realidad virtual es la menor, pero no la menos importante, de las disciplinas que intervienen en el trabajo cooperativo. Un sistema de realidad virtual es un sistema interactivo usado para crear un mundo artificial o sintético en el cual el usuario tiene la impresión de estar presente, navegar y manipular al resto de los objetos[13]. Existe una clasificación básica para este tipo de sistemas:

1. Sistemas Inmersivos
2. Sistemas No Inmersivos

Los sistemas inmersivos tienen por objeto conseguir que el usuario tenga la sensación de estar presente en el mundo artificial. Esto se logra mediante los dispositivos que impiden la visión del mundo circundante, mientras que presentan las imágenes del mundo virtual. El ejemplo más claro son los HMD (Head Mounted Display) o visores[13].

Por parte de los sistemas no inmersivos, existen distintos grados de proyección en estos sistemas, algunos están basados en el hecho de que el usuario se introduzca en una habitación o adminículo cerrado en cuyas paredes se proyectan una o más imágenes del mundo virtual. Con un grado menor de inmersión se tienen clasificados a aquellos que presentan las imágenes en las pantallas de las computadoras.

Estructura de los sistemas de Realidad Virtual (RV)

La estructura básica de un sistema de RV contiene 3 elementos importantes[25]:

1. Dispositivos de Entrada
2. Dispositivos de Salida
3. Dispositivo de Simulación y Control

Mediante éstos, el sistema de RV construye los puntos de interacción hacia el usuario, es por ellos, que la información fluye bidireccionalmente respetando el protocolo que especifica el modelo particular en una aplicación dada.

Los datos que requiere la computadora para calcular el entorno presentado por los dispositivos de salida son:

1. Posición de cada objeto dentro del mundo
2. Posición del usuario
3. Una referencia del objeto en el que el usuario se interesa, cada vez que actúa mediante los dispositivos de entrada

En una aplicación sencilla de RV podemos observar los siguientes pasos básicos durante su ejecución:

1. Lectura de los dispositivos de entrada. Se lee información desde el dispositivo de entrada.
2. Detectar los objetos de interés. Se realiza el cálculo y la simulación delmundo virtual.
3. Actualizar la posición del usuario. Actualizarla relativamente a la posición de los otros objetos. Determinar si algún objeto debe cambiar de acuerdo a las acciones del usuario.

4. Enviar la imagen al dispositivo de salida

A continuación mencionaremos los tipos de dispositivos de entrada y salida mas comunes en los sistemas de RV.

Dispositivos de Entrada

Las interfaces existentes al inicio de la década de los años 70, se hicieron insuficientes ante el nacimiento de los primeros simuladores, por lo que se diseñaron dispositivos que fueran mas útiles y apropiados a este tipo de sistemas.

Estos se encargan de proporcionar al usuario las capacidades de navegación, interacción y manipulación en el ambiente y de notificar al sistema sus acciones.

Estamos hablando de dispositivos como:

1. Guantes sensibles
2. Sistemas de reconocimiento de voz
3. Sistemas bioeléctricos
4. Ratón 3D y track-ball

Los guantes sensibles se emplean en aplicaciones con una mano virtual y de telerobótica. Están diseñados a base de sensores de flexión mecánicos o de fibra óptica de tal forma que los movimientos detectados son transmitidos a la computadora y posteriormente recreados.

Los sistemas de reconocimiento de voz generalmente necesitan adiestramiento, entre 300 y 1000 frases y sus repeticiones habilitan a estos programas para que reconozcan ciertos patrones en la voz de una sola persona.

Por otro lado, los sistemas bioeléctricos se basan en el procesamiento de señales bioeléctricas del ser humano con el fin de monitorizarlas e interpretarlas en el sistema. La empresa Biocontrol Systems de California,

diseñó entre otras aplicaciones, dos sistemas importantes; uno llamado Eyecon [42]basado en la posición de la mirada del usuario pretendía sustituir al mouse y / o poner esta tecnología al alcance de personas que no pueden utilizarlo. El sistema Brainman[42] detecta las ondas cerebrales y calificándolas podía disertar aproximadamente cual era la condición mental, estable o no, del usuario.

El ratón 3D y los track-ball son instrumentos de precisión que permiten mas grados de libertad en el espacio tridimensional, normalmente se usan en telerobótica, por ejemplo, en simuladores que enseñan a las personas como conducir un vehículo de carga pesada o para la construcción, o bien, un submarino o un transbordador.

Dispositivos de Salida

Son aquellos en los que el usuario ve el mundo presentado por el sistema, por ejemplo:

1. Pantallas de proyección
2. Head Mounted Display (HMDs) o visores
3. Sistemas binoculares

Las pantallas de proyección y monitores de computadora, son el medio mas común y al alcance de las personas.

Los visores evolucionaron en los últimos 20 años, desarrollándose con tecnologías de tubos de rayos catódicos (CRT) hasta pantallas de cristal líquido (Liquid Crystal Display, LCD) monocromo o en color, la utilización de fibra óptica para enviar los datos desde el casco a la computadora, auriculares estereofónicos y sensores de posición. La resolución de estos dispositivos está al rededor de 400 x 300 pixeles y los ángulos de visión mas comunes son en sentido horizontal, 110° y en sentido vertical de 90°

Los sistemas binoculares son de mayor tamaño y mucho mas caros, su resolución está en el orden de 1000 x 1000 pixeles y están formados por un pie que sostiene una barra, en la cual un extremo sostiene el dispositivo de proyección y el otro tiene un fuerte contrapeso para dar la sensación de fuerza y resistencia durante la navegación.

Características de los Sistemas RV

Cualquier sistema de software que cumpla con las siguientes características[23] podrá ser correctamente calificado como sistema de RV:

Capacidad sintética. La síntesis consiste en la generación de las imágenes y escenas en tiempo real, esto es, el escenario se crea en el momento que el usuario comienza a interactuar con los dispositivos de entrada del sistema pues resulta imposible calcular y almacenar de antemano las imágenes correspondientes a todas las posibles posiciones. Existen en la actualidad numerosos sistemas que permiten ver imágenes de video en tres dimensiones, pero eso no es RV porque se trata de imágenes pregrabadas, es decir, no se sintetizan en tiempo real.

Interactividad. La capacidad de síntesis no tiene sentido si no se permite al usuario actuar en el mundo virtual. Si el usuario no tuviera la posibilidad de influir sobre el estado del mundo de ninguna forma, no sería necesario sintetizar las imágenes, porque bastaría con promover un paseo virtual. Un parámetro fundamental de los sistemas interactivos es el de la velocidad de respuesta (latency), es decir, el tiempo transcurrido entre el instante en que el usuario efectúa una determinada acción y el momento en el que el sistema actualiza la información de salida que entrega al usuario.

Tridimensionalidad. Esta característica explota la capacidad del ser humano de reconocer espacios de tres dimensiones.

Ilusión de realidad. La ilusión de realidad se intensifica cuanto más fantástico es el mundo virtual, esto es, cuantas menos equivalencias con el mundo real, menos referencias tiene el usuario para cuestionar la validez de lo que percibe. Sin embargo, las reglas del mundo virtual deben ser entendibles al usuario.

Mecanismos básicos y herramientas empleados en aplicaciones de RV

Los mecanismos mas comunes utilizados para el desarrollo de este tipo de sistemas son:

Gráficos tridimensionales. Las técnicas utilizadas en el tratamiento de gráficos 3D se han desarrollado a partir de los años 70, en la Universidad de Utah. En general, sabemos que se trata de un proceso de transformación de figuras básicas en función de la posición del usuario y que el resultado es una imagen de la escena alterada en sus propiedades de perspectiva, iluminación, reflexión, etc.

Técnicas de estereoscopía. Estas técnicas permiten al usuario no sólo percibir las claves de profundidad, sino además ver las imágenes efectivamente en relieve. Esto se logra gracias a que se proyecta una imagen distinta para cada ojo y, por las diferencias relativas, el cerebro del usuario interpreta cierto relieve en las figuras.

Simulación de comportamiento. Esta tarea tiene dos cometidos importantes:

1. Cuando se emplea para remedar el comportamiento de algún objeto o sistema del mundo real.

2. Donde los objetos simulados no tengan un equivalente demasiado directo en el mundo real, la apariencia de realidad del sistema vendrá

dada por factores subjetivos, como variedad, coherencia y un alto grado de interactividad.

Facilidades de navegación. Se trata de los mecanismos puestos a disposición del usuario para poder variar su posición u orientación dentro de una escena. Esto es, dispositivos de control y dispositivos de localización.

Técnicas de inmersión. Como se mencionó con anterioridad, debemos aislar al usuario de los estímulos procedentes del mundo real circundante, con el fin de acentuar la apariencia de realidad de la aplicación. De tal forma que el usuario pierde la referencia con la que poder comparar las sensaciones que el mundo virtual produce.

Herramientas de Programación. Para diseñar los mundos virtuales se dispone actualmente de una amplia lista de shareware y software especializado, por ejemplo el lenguaje VRML (Virtual Reality Modeling Language) del consorcio Web3D[38][43][44][45], que es considerado el lenguaje oficial de la realidad virtual, 3D Studio Max[50], TrueSpace[51], entre otros. Podemos mencionar también una serie de paquetes especializados que proporcionan herramientas útiles al desarrollar ambientes virtuales bajo el estándar de juegos de estrategia o multiniveles, tales como 3DGPL[52], CyberTown[55], DIVE[47], DIVA[56], Fly3D[57], Genesis3D [58]y Legus[59].

2.4 Antecedentes

En este punto debe de quedar claro que en la bibliografía relacionada al tema, no fue encontrado ningún trabajo que tuviera nuestro objetivo. Por lo que decidimos dividir nuestro trabajo en una revisión de las áreas de

informática involucradas para adquirir los conocimientos y fundamentos necesarios a nuestra empresa y, por otro lado, hacer una revisión de la clase de sistemas existentes en esta línea de investigación. Esta revisión en conjunto, fue usada para definir los requerimientos de nuestra Arquitectura Genérica. En las secciones anteriores hemos resumido la revisión de las áreas de la informática involucradas en nuestro proyecto[2] que soportará la implementación de una gran diversidad de sistemas semejantes a los estudiados. A continuación, haremos una resumen de la literatura que trata de sistemas afines, su funcionamiento y posteriormente un análisis de las herramientas que utilizaron en su desarrollo.

2.4.1 Sistemas afines

Entre los sistemas distribuidos cooperativos que encontramos hay una gran parte que están dedicados al desarrollo de mundos virtuales y a la creación de oficinas virtuales, en ambos el objetivo es crear un ambiente que sea lo más semejante al de los mundos reales.

Los diseñadores consideraron como importantes distintos aspectos. Algunos se preocupan por los aspectos administrativos o estratégicos de la empresa, y se auxilian de sistemas de RV para alcanzar objetivos puramente relacionados con la posición en el mercado de dicha organización[62]. Otros están concentrados en aspectos relacionados con CSCW[1]; otros más se dedican al problema de mantener una adecuada consciencia de la información[23].

No todos los sistemas utilizan audio y video, sin embargo, siempre fue una característica recomendable. En otros casos[63][48], se facilita el trabajo cooperativo a través de un sistema de video conferencias, de esta manera pone particular atención en la comunicación informal[64]. El trabajo de

[2]En realidad existen muchas otras áreas involucradas en el diseño de una herramienta génerica, por ejemplo interfaces hombre máquina, ergonomia, inteligencia artificial, sin embargo en este estudio nos límitamos a aquellas que tenian incidencia en nuestro trabajo de generar una arquitectura genérica.

Montage[?] maneja conceptos de teleproximidad y usa video como principal medio de comunicación entre los colaboradores. DIVA[56] [60]tiene gran prestigio y es muy conocido en este medio, es un ambiente de oficina virtual que soporta comunicación, cooperación y consciencia de la información en modo síncrono o asíncrono. FreeWalk[61], trabaja con un mecanismo de control muy hábil de la posición y orientación de los usuarios. Massive[62] se basa en el control de la posición del foco de atención del usuario, de su posición, y de una región que llama aura de interacción.

El trabajo realizado por Shinkuro Honda et all[55] fue una fuente importante para ponderar nuestro proyecto. Realizaron estudios que analizan el comportamiento de las personas al trabajar individualmente y en equipo, y distinguieron algunos rasgos en la conducta que ayudaron a definir si el sujeto estaba concentrado o no, esto con el objeto de que el sistema reaccionara adecuadamente y lo mantuviera atento. Su investigación está completamente dirigida a la forma en que el usuario debe recibir la información, la magnitud y complejidad de la misma, para no agobiar al empleado y hacerlo sentir como en un espacio real de trabajo, lo que nos permitió precisar los alcances de nuestro proyecto.

Otro proyecto de investigación[63] presenta la arquitectura de un sistema distribuido multiusuario con un ambiente virtual desarrollado en un laboratorio de percepción virtual en los EU y además muestra un dispositivo de navegación especialmente diseñado para la utilización de este proyecto. La especificación de este sistema se realizó con el lenguaje Linda en su versión para Prolog. Linda es un lenguaje independiente de la máquina, de programación paralela que permite la creación de bancos de información compartida, diseñado en la universidad de Yale en 1989.

Roehl tiene información sobre Realidad Virtual Distribuida[64], este trabajo desarrollado en 1994 está orientado a explorar los fundamentos de la implementación de las redes de computadoras y de protocolos como DIS (Distributed Interactive Simulation Protocol) y la familia

TCP/IP (Transmission Control Protocol / Internet Protocol) analizándolos y sugiriendo que ventajas pueden aplicarse al desarrollo de sistemas distribuidos de RV. En este mismo año, crea una interesante estructura taxonómica[65] de la entidades de los sistemas de RV en función de su comportamiento y una escala de niveles de conducta de acuerdo a la complejidad de la información que éstas manejan.

2.4.2 Funcionamiento de sistemas afines

En la sección anterior se resumió una lista de sistemas que consideramos importantes para especificar nuestra propuesta. Nos dimos cuenta de que hay muchos ejemplos de sistemas distribuidos cooperativos que contemplan, espacios compartidos de información y ambientes 3D. De esta gama, seleccionamos tres que contienen información crítica que influyó en nuestras decisiones. Estamos hablando de los aspectos más significativos del sistema de Shinkuro Honda[55], Sohlenkamp (DIVA)[56] [60] y los conceptos de Roehl[64][65].

Shinkuro Honda et all

Como mencionamos, el sistema de estos investigadores proporciona un ambiente de trabajo cooperativo distribuido llamado oficina virtual, en donde los multimedios (texto, audio y video) son el canal de comunicación utilizado. Dentro de los estudios que fueron realizados está la evaluación de los aspectos reales de la comunicación en grupos de personas. Los resultados de esta evaluación son que en donde es muy útil mantener un ritmo de trabajo apropiado existe un intercambio de información abundante a través de la comunicación formal e informal.

En el sistema distribuido un factor importante para soportar estos tipos de comunicación es proporcionar el sentimiento a los miembros que se encuentran realmente en el ambiente virtual a esto se le llama consciencia

de la información o en inglés *"awareness information"* Satisfacer este requerimiento es muy complejo, puesto que va desde visualizar a todos los elementos hasta distinguir emociones u estados de ánimo entre los miembros presentes. Desde el punto de vista técnico, este requerimiento hace necesaria una constante transmisión de la información que mantenga a cada usuario consciente del estado en que se encuentran los otros.

En este sistema para evitar comunicaciones inútiles, es decir más información que la necesaria para entender el estado del ambiente, es delimitado el espacio de información que debía transmitirse a cada usuario, de tal forma que sólo recibiera los estímulos que se efectuaban a su alrededor o bien, aquello que le interesaba del ambiente. Por lo que definieron el concepto de *espacio personal* y *grados de concentración.* Esto significa que cada empleado en la oficina puede delimitar un área radial con la que muestra cierta sensibilidad hacia el exterior. Por ejemplo, una persona que desea concentrarse puede cerrar esta área, de tal forma que para recibir información del ambiente, debe tratarse de hechos que se realizan de forma cercana a ella. Mientras que una persona que desea estar consciente de los eventos que le rodean solo tendría que abrir este espacio para recibir audio y video de los otros participantes.

Para crear esta consciencia entre los miembros proporcionaron dos conceptos llamados Vista circundante (Around view) y Efectos de sonido (Sound Effects).

Vista Circundante. En el mundo real, cualquiera es capaz de reconocer si una persona esta a un lado, de pie o sentado, sin mirar. Esto es debido a que el campo de visión de un ser humano se extiende por 180 grados. Sin embargo, si en la representación tridimensional del mundo se pudieran observar esos 180 grados, no sería necesario girar para estar consciente de muchas cosas, y por otro lado, habría confusión en cuanto a la orientación de la figura que representa a cada cliente del ambiente y sobre a quién se está dirigiendo al interactuar.

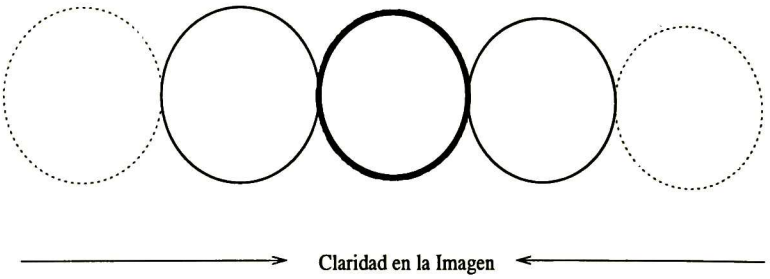

Figura 2.1: Ejemplo de la distribución de la visión en el sistema de Shinkuro et all

El humano puede reconocer claramente lo que ve en 60 grados y en el resto existe cierta distorsión mientras se abre el cambio de visión, ésta era la respuesta para la forma en que el ambiente debía mostrarse para causar la sensación de realidad. Para ilustrar lo descrito observe la Figura 2.1,en donde las formas en los extremos son difusas con respecto a las que se presentan en el centro.

Con este efecto implementado, intentaron lograr un grado avanzado de realidad en el sistema.

Efectos de Sonido. Los tipos de sonido transmitidos en este ambiente son de caminar, de sillas y puertas entre otros; todos fueron implementados con el objetivo de aumentar la sensación de realidad. Su funcionamiento se basa en la transmisión controlada de sonido en el medio estéreo, para que el usuario detecte cierta posición en su origen.

Implementación

La arquitectura del sistema ilustrada en la Figura 2.2, esta basada en el paradigma de cliente y servidor. Los roles de cada componente son:

- LoginServer. Administra la entrada y salida de cada miembro. Al conectarse a este servidor cada cliente puede obtener información de otros usuarios.

- LocationServer. Este indica la posición de los otros miembros,

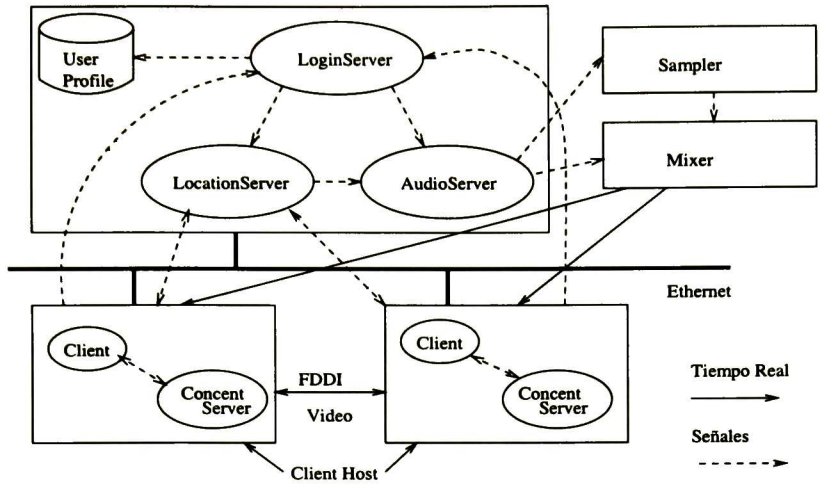

Figura 2.2: Arquitectura del sistema de Shinkuro et all

su dirección y grado de concentración, valores que cambian constantemente.

- AudioServer. Controla los efectos de sonido con la información posicional del LocationServer.
- Client. Esta es la aplicación que despliega la oficina virtual, muestra la localización precisa de los otros miembros gracias a la información proporcionada por el LocationServer.
- ConcentServer. Este servidor se ejecuta por cada cliente para recalcular su grados de concentración por periodos de un minuto y envía los resultados a Client.

Bernie Roehl

Los trabajos revisados de este autor son importantes a nivel conceptual y práctico.

En el documento llamado "Some Thoughts on Behavior in VR

Comportamiento Deterministico	Comportamiento Indeterministico
Estaticas	Newtonianas
Animadas	Inteligentes

Figura 2.3: Categorías de sistemas de realidad virtual distribuida

Systems"[66], Roehl describe que si una entidad está definida por medio de un conjunto de atributos, el comportamiento está definido por el cambio de estado de cada uno de estos atributos. Tomando como base el comportamiento así definido de las entidades de un sistema, el autor define la siguiente una estructura taxonómica basándose en su comportamiento determinístico y no determinístico. Determinístico significa que podemos definir el estado de la entidad en un periodo de tiempo específico. Indeterminístico significa un comportamiento naturalmente impredecible. Estos tipos básicos de entidades son a su vez subdivididos como es ilustrado en la Figura 2.3. Las principales características de los elementos de la taxonomía definida son:

Entidades estáticas. Aquellas de las cuales sabemos que su estado nunca cambia y por tanto, este estado es siempre conocido en cualquier momento.

Entidades animadas. Este tipo de componente cambia su estado constantemente. Los cambios son fácilmente predecibles pues son una función del tiempo y de algunos otros parámetros predefinidos.

Entidades newtonianas. Estas entidades responden a los cambios de su ambiente reaccionando de acuerdo a las leyes físicas implementadas por los creadores.

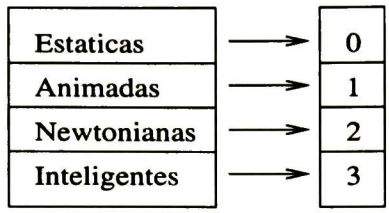

Figura 2.4: Comparación entre los tipos de entidades y los niveles de comportamiento de Rohel

Entidades inteligentes. Aparentan tener voluntad propia, tienen objetivos específicos, un comportamiento complejo y son inherentemente impredecibles.

Ahora bien, todas estas entidades pueden tener además distintos niveles de comportamiento:

0. Modificación directa de los atributos de la entidad
1. Cambio en los atributos de la entidad a través del tiempo
2. Conjuntos de funciones de nivel 1 para realizar una tarea
3. Toma de decisiones para realizar una tarea ejecutando varias instrucciones de nivel 2.

Concluye mostrando una relación entre los distintos tipos de entidades y los niveles de comportamiento, ilustrando con esto los alcances de todos los posibles elementos de un sistema de realidad virtual distribuido, tal como se muestra en la Figura 2.4. Bernie Roehl en su trabajo titulado "Distributed Virtual Reality - An Overview" se basa en el protocolo DIS (Distributed Interactive Simulation Protocol) utilizado en el ejército para construir algunos simuladores distribuidos, aunque éste no está precisamente dirigido al estudio de la realidad virtual distribuida, sí se enfrenta a los mismos retos que se presentan ante un desarrollador de nuestra línea de estudio.

La actualización de la información del mundo virtual en todos los clientes del sistema es uno de los principales problemas a resolver por coherencia de

la información. Roehl considera que esto puede ser realizado con broadcast o multicast.

Realizarlo por medio de broadcast puede ser desde un servidor dedicado o bien, desde cada uno de los clientes. Por otro lado, empleando multicast, es decir, aplicando un filtro en la actualización de los mensajes entre los componentes, un servidor controla que clase de información se enviará a cada cliente, aunque es claro que esto podría ser realizado desde el programa de cada usuario.

La especificación de DIS, tiene ciertas ventajas y desventajas que el autor menciona como útiles, en general, para el desarrollo de sistemas de realidad virtual distribuida. DIS proporciona un formato estándar en los mensajes, un modelo completamente descentralizado, áreas de interés en la distribución de la información, uso de multicast. No obstante, DIS ha sido diseñado para simulación militar, una aplicación muy específica que no podría adaptarse fácilmente a un sistema distribuido con objetivos distintos, por ejemplo, en el entretenimiento del usuario.

DIVA de Sohlenkamp

DIVA es un prototipo de un ambiente de oficina virtual que proporciona las operaciones de comunicación, cooperación y consciencia en los modos síncrono y asíncrono. Integra en una interfaz simple e intuitiva que pretende reemplazar la clásica interfaz de usuario. Este sistema presume de aportar todas las herramientas y facilidades a los usuarios de distintos y numerosos equipos de trabajo.

Su herramienta ha sido diseñada para actividades de CSCW. En general, integran un conjunto de tecnologías de trabajo en grupo en una sola interfaz, a fin de eliminar las muchas veces inevitables interrupciones; además, analizaron la intuición, conocimiento y habilidades de las personas que tienen experiencia trabajando en equipo.

En la tabla de la Tabla2.1 observamos 6 clases de actividades que son

Funcionalidad/Tiempo	Síncrono	Asíncrono
Comunicación	Comunicación en Tiempo Real	Escribir notas para otros
Cooperación	Trabajo simultáneo usando herramientas de groupware	Trabajo por turnos
Consciencia	¿Que hacen los demás?	¿Qué han hecho los demás recientemente?

Tabla 2.1: Las 6 clases conceptuales que definen las actividades en la Oficina Virtual de Sohlenkamp

contempladas por este sistema:

Este marco conceptual clasifica precisamente las funciones de un ambiente distribuido para trabajo cooperativo.

Los elementos de esta oficina virtual son personas, habitaciones, escritorios y documentos:

Personas. Se mueven a través del ambiente y tienen asignada una habitación dentro de la oficina. Se puede determinar que están realizando en cualquier momento y con quienes tienen abierto un enlace de audio y video.

Documentos. Son en realidad, archivos de texto, imagen, audio y/o video con que las personas trabajan en el ambiente, pueden ser consultados y/o modificados desde distintos lugares de la oficina virtual.

Escritorios. Lógicamente, reúnen a un grupo de personas reducido que trabajan sobre los mismos documentos. Normalmente, existen varios escritorios en cada habitación de la oficina.

Habitaciones. Albergan a las personas, escritorios y documentos. Controlan los enlaces de audio y video de los usuarios, de forma que, cuando un usuario entra en alguna habitación virtual que está ocupada

> entonces se establecen los enlaces de video y audio entre los ocupantes y el recién llegado. Estas habitaciones, son utilizadas como oficinas privadas, salas de reunión o espacios de propósito especial.

Cuando los autores se refieren a que esta interfaz puede desarrollar actividades en modo síncrono y asíncrono se refieren al tipo de servicios que ofrecen. Por ejemplo, si se trata de la comunicación persona a persona en tiempo real en una conferencia con audio y video entonces se trata de comunicación síncrona. El servicio de las notas (stick-on notes) en documentos, escritorios, usuarios y oficinas es un ejemplo de la forma asíncrona de trabajo, pues estas notan guardan comentarios entre personas que no se encuentran en un mismo tiempo en la oficina.

Desde nuestro punto de vista, esta interfaz es amigable y contiene herramientas muy útiles de comunicación y para la actualización del estado del espacio compartido y nos orientó con respecto a la diversificación y clasificación de las operaciones que los usuarios pueden realizar en este tipo de sistemas.

2.4.3 Herramientas

En esta sección hacemos una descripción de las herramientas utilizadas en algunos de los sistemas que consideramos relevantes. Este análisis nos proporciono valiosos criterios para escoger aquellas que nosotros emplearíamos para implantar tanto la arquitectura como el prototipo.

En la tabla2.2 mostramos los instrumentos tecnológicos empleados por Shinkuro et all[55] en la implementación de su oficina virtual.

Si hubiéramos tratado de emplear esta misma tecnología nos hubiéramos enfrentado a altos costos en cuanto a la adquisición del equipo y licencias del software. El lenguaje de programación g++ normalmente está integrado al ambiente del sistema operativo y hubiera sido fácil utilizarlo, sin embargo

Característica	Recurso
Estación de trabajo	Silicon Graphics Indy R5000 O2 R10000
S.O.	IRIX 5.3
Sistema de ventanas	Indigo Magic
Lenguaje de programación	g++ Versión 2.7.2
Interfaz de usuario	OSF/Motif, OpenGL
Otras Herramientas	FDDI LAN (cisco) mezclador DMP11 (YAMAHA) sampler EPS16-plus (Ensoniq) PC9801 (NEC)

Tabla 2.2: Arquitectura del Sistema, Shinkuro Honda, Japón

debido a nuestro requerimiento de generar un sistema multiplataforma las posibilidades de que se lograra se reducen considerablemente. En el desarrollo de la interfaz, Motif requiere licencia, mientras que OpenGL es un estándar abierto y gratuito, de hecho, esto nos indicó que era una excelente opción para nuestro trabajo. Por la parte del desarrollo de audio y video no coincidieron nuestras metas, pero estas opciones se consideran como viables en realización del que hoy es trabajo futuro. El tipo de red en el que trabajaron debió influir en el aspecto de acción reacción de su sistema, en nuestro caso, podemos afirmar que se tiene un rendimiento aceptable, sabiendo que nuestros recursos son menores que los de una red FDDI, entendemos que finalmente nuestro sistema tendrá un buen desempeño.

El trabajo de Sohlenkamp et all[40] con DIVA es muy impresionante en cuanto a la cantidad de servicios que ofrece. Las herramientas utilizadas son las mostradas en la tabla 2.3.

Cuando un grupo de usuarios modifica un sólo documento o participa activamente sobre un recurso compartido, es necesario activar algunos mecanismos de sincronización y mantenimiento de la información de la interfaz. Estos mecanismos se implementaron con el software GINA en

Característica	Recurso
Lenguaje de programación	Common Lisp and CLOS (Common Lisp Object System)
Platfomas de Red	Ethernet y ATM
Otras herramientas	MultiUser GINA Framework OSF/Motif software, C++

Tabla 2.3: Arquitectura del Sistema, Markus Sohlenkamp et all, Alemania

Common LISP[3], y esto facilitó la implementación de la oficina virtual presentada por Sohlenkamp. El equipo de trabajo del proyecto DIVA/GINA está formado por 6 integrantes, 4 desarrolladores y los autores como líderes de proyecto. Sus pruebas fueron realizadas sobre redes con Ethernet y aseguran que dado que el principal motivo de posibles deficiencias son debidos a los retrasos de red, los resultados deben ser más satisfactorios.

En comparación con las expectativas de nuestro proyecto, podemos decir que estas herramientas no son muy útiles en el plano multiplataforma que debemos respetar, sin embargo, hemos aprovechado los conceptos de trabajo en grupo, comunicación y consciencia de la información que nos proporcionó su investigación.

2.5 Conclusión

En este capítulo describimos gran parte de la investigación realizada:

En primer lugar presentamos la revisión efectuada sobre los conceptos fundamentales de nuestra tesis que se refieren a aspectos de sistemas distribuidos y conceptos básicos relacionados con el tema de trabajo cooperativo.

En segundo lugar, se presento un resumen de los aspectos importantes de

[3]GINA se puede emplear con dos implementaciones, una para trabajar con Common Lisp y otra para C++.

implementaciones de sistemas de trabajo cooperativo y de las herramientas que los diferentes equipos de trabajo utilizaron en la implantación de sus sistemas.

Al dividir nuestra revisión del estado del arte, nosotros estamos: 1. Asegurando que nuestro aporte al estado del arte es original, en cuanto a que no existe un trabajo similar para el desarrollo de una plataforma genérica, y 2. Haciendo una revisión de los sistemas más sobresalientes desde nuestra perspectiva, en cuanto a CSCW, porque nos permitieron establecer los criterios necesarios para llevar a cabo un adecuado diseño e implementación, mismos que serán descritos en los siguientes capítulos.

Ahora bien, de acuerdo a todo esto, la arquitectura que nosotros proponemos puede presentar algunos tipos de fallas expuestos, aunque hemos implementado los mecanismos para evitarlas en lo posible; se trata de un sistema distribuido asíncrono que soporta la integración de otros sistemas distribuidos que pueden ser síncronos o asíncronos, como se explicará en el capítulo 3.

Para terminar, debemos decir que hemos adoptado la clasificación de Roehl porque creemos que abarca de forma completa todos los papeles que pueden asumir una entidad de un sistema distribuido con realidad virtual.

Capítulo 3

Arquitectura GeDA-3D

3.1 Objetivo

Al termino este capítulo conoceremos los componentes y principales características de nuestra Arquitectura Genérica, la cual constituye la base para definir aplicaciones CSCW distribuidas.

3.2 Introducción

Como lo mencionamos anteriormente, un sistema de trabajo cooperativo es un sistema que ayuda a un conjunto de entidades humanas e informáticas cooperar para lograr un objetivo. En la literatura a la plataforma informática se le denomina Groupware[31].

En nuestro caso, cuando nos referimos a una Arquitectura Genérica, hablamos de una capa de software que brinda un conjunto de características constantes utilizables en el desarrollo de sistemas distribuidos de naturaleza cooperativa.

Nuestra propuesta adopta un esquema de coordinación-control ortogonal, es decir, que las actividades de control son desempeñadas por agentes distintos de aquellos que realizan las funciones específicas del sistema.

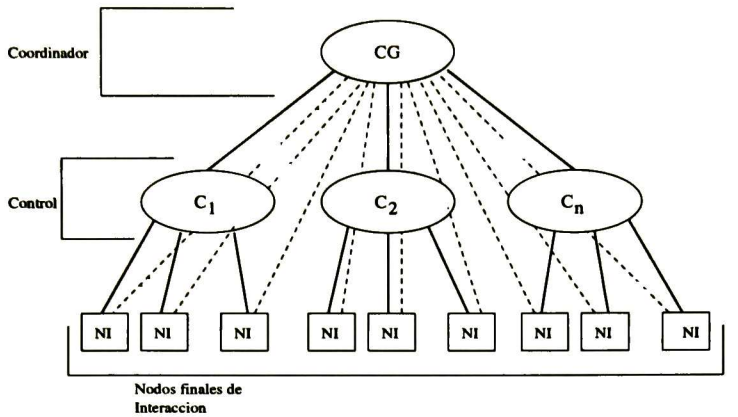

Figura 3.1: Esquema de coordinación

Esta decisión, ayuda a determinar una estructura base, la cual muestra una independencia entre las operaciones de comunicación y coordinación y aquellas que están relacionadas con el tipo de aplicación que se implementa. Tanto las funciones de coordinación y comunicación como las de la aplicación, tienen fronteras definidas como se muestra en el esquema de la Figura 3.1.

Un punto importante a decidir fue qué paradigma utilizar para diseñar nuestra arquitectura. En base a los requerimientos obtenidos de nuestro objetivo y la investigación en la primera fase de desarrollo, decidimos hacer uso del paradigma de agentes software[67]. Muy brevemente, y para nosotros ya que no existe una definición estándar:

Un agente de software es un programa autónomo que puede realizar una o varias tareas a las cuales se les conoce como competencias.

Tomando como base ésta, nuestra definición, podemos describir nuestra arquitectura como una comunidad de agentes, formada por:

Agente Coordinador General. El objetivo principal de este agente es la administración del sistema generado por el desarrollador. Para realizar esta tarea el agente ofrece a los desarrolladores y a los usuarios finales el siguiente conjunto de funciones:, los medios que necesitan para llevar a cabo sus actividades de forma natural:

***Adscripción** de entidades de control.* El agente CG dedicará toda su capacidad de procesamiento a las funciones de seguridad o control de acceso de los usuarios al sistema, mantener las bases de datos (BD) que contienen la información de cada uno de ellos y mantener el estado único compartido del ambiente tridimensional además de distribuirlo estas entidades constituyen los puntos de procesamiento de las funciones o servicios que se proporcionarán dentro del ambiente 3D. Estos componentes deben cumplir con algunas condiciones para poder adaptarse. Una vez integrados tienen una representación gráfica y con ello, el medio de activación con el que se mostrarán disponibles al usuario.

***Control** de usuarios.* Esta actividad es enteramente controlada por el agente CG, siempre que un usuario ingrese al sistema, el CG validará su entrada consultando la información de la BD de usuarios, en donde también pudiera encontrar sus datos generales y derechos que hayan sido negociados previamente. Esto no significa que las entidades de control o servicios no pudieran implementar también sus propias funciones de registro y control de usuarios. Integración a un solo sistema de software. Es una ventaja para el usuario y su sistema operativo tener solamente una aplicación que proporcione todos los mecanismos para cumplir con una tarea completamente. De esta forma, se eliminan algunos cambios de contexto que pueden distraer al usuario, que debe palpar principalmente, esta realidad virtual.

***Interfaz** gráfica 3D.* El ambiente tridimensional es independiente de cualquier navegador de páginas web en la internet, los usuarios conectados al sistema pueden verse unos a otros como iconos volumétricos en un escenario compartido que puede ser cualquier ambiente imaginable. Aquí cada uno encontrará objetos con propiedades estáticas, con las cuales un objeto solo participa en el

escenario con fines ornamentales, o propiedades dinámicas, debido a que con ellas la figura permite iniciar la ejecución de una función variable llevada a cabo por un elemento de control del modelo de coordinación. Cuando esto sucede, el ambiente virtual abre paso a la aplicación específica y se reinicia este procedimiento cuando termina.

Globalización. Lo que el modelo ofrece es una plataforma que trabaja con la familia de protocolos TCP/IP[28], por lo tanto, existe un inmenso número de personas que pueden usar este concepto o implementación. No es necesario más que una conexión hacia el internet para instalar un sistema completo de este tipo o bien, un sistema público que internacionalice las aplicaciones adscritas.

El agente CG mantiene abierto un canal de comunicación con los agentes de control distribuido ($\kappa = \{C_1, C_2, ..., C_n\}$). Este canal constituye el medio para la transferencia de datos entre cada componente. Las acciones de cada elemento de este conjunto κ, por su parte, son definidas en función de la aplicación, es decir, son del tipo de software que se está desarrollando, lo que proporciona flexibilidad al modelo.

Agentes de control. Están en constante comunicación con el coordinador general, realizan tareas específicas relacionadas con la naturaleza de la aplicación. Su creación puede ser independiente de las otras entidades.

Agentes de Interacción (AI) o nodos. Son la ventana de comunicación entre los usuarios y el sistema, intercambian información con el coordinador general y las entidades intermedias, lo cual influye en la línea de ejecución del programa. En general se encargan del mantenimiento de la interfaz tridimensional, es a través de ellos que el sistema recibe entradas y muestra resultados. Hay distintos tipos de AI , lo cual se determina en función de la participación de los usuarios y de la clase de perfiles que tienen. Por ahora, hablaremos de dos perfiles de usuario importantes que son los desarrolladores y usuarios. Sabiendo que todo este proyecto está dirigido a

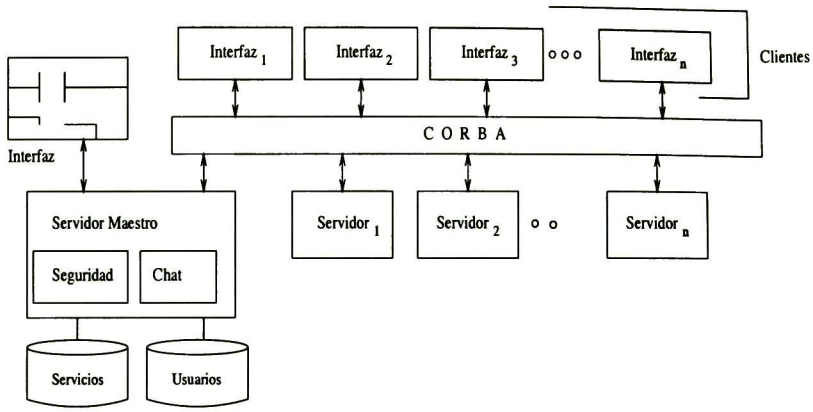

Figura 3.2: La primera aproximación

satisfacer sus necesidades era necesario determinarlos en las primeras fases de ingeniería. Los desarrolladores utilizan este modelo como plataforma de integración de un conjunto de sistemas distribuidos. Los usuarios finales utilizan la aplicación creada por el desarrollador a partir de nuestra base conceptual y de implantación. Por tanto, es posible identificar tres tipos de nodos, el primero corresponde a los desarrolladores, el segundo sera empleado por usuarios, finalmente el tercero permite a un desarrollador ser administrador del sistema.

Con base en esta primera especificación informal, la Figura3.2 ilustra nuestra primera aproximación de Arquitectura Genérica.

Esta arquitectura muestra un esquema modular, en donde ya están definidos los roles principales de nuestra abstracción de un sistema distribuido genérico. Entre las competencias del agente Servidor Maestro están:

- Coordinar todas las actividades relacionadas con la conexión, el registro, el uso del ambiente y el control de los servidores dedicados de la aplicación.

- El agente Servidor Maestro es el encargado de la consistencia del medio ambiente, indispensable en la implantación de un sistema CSCW

eficiente. El agente Servidor Maestro implementa la consistencia, a través una distribución de cualquier modificación del medio a todos los clientes que se ven afectados.

3.3 Arquitectura Genérica GeDA-3D

La arquitectura GeDA-3D ha sido diseñada con el objetivo de facilitar la implantación de sistemas distribuidos cooperativos. De manera muy simple, GeDA-3D integra una interfaz amigable y confiable con n sistemas distribuidos de naturaleza indistinta.

La arquitectura GeDA-3D está conformada por un agente Servidor Maestro, un agente Cliente Administrador, n clientes/servidores agregados, n clientes maestros y n clientes agregados. A continuación se definen las competencias de cada agente.

3.3.1 Competencias de la comunidad de agentes

El agente Servidor Maestro (*SM*): Este agente es la implementación del CG del esquema de coordinación especificado en la sección precedente. Las competencias de este agente son:

- Administración de servidores dedicados o agentes de control.
- Control y administración de usuarios
- "Seguridad"[1]
- Consistencia del estado compartido del ambiente virtual.
- Administración de las bases de datos que contienen información sobre los usuarios y agentes de control.

El Agente Cliente Maestro (CM): Cada usuario posee una aplicación cliente a la que llamamos *Agente Cliente Maestro* que se ejecuta localmente. El agente CM permite al usuario después de haber pasado del proceso de conexión y validación, usar el módulo de comunicación informal (chat) y el ambiente virtual 3D. Dentro del diagrama mostrado en la Figura3.1 del modelo de coordinación, este elemento corresponde a los agentes de interacción o nodos finales del diagrama de coordinación utilizado, puesto que es por medio de ellos que el usuario interactúa con el sistema.

Los Agentes de Servicio (AS): Dentro del esquema de coordinación mostrado en la Figura3.1, estos agente están asociados a los elementos del conjunto κ de agentes de control. Debe notarse que no ha sido establecida ninguna hipótesis acerca de estos agentes AS, es decir, que dado que éstos son de clase variable, proporcionan flexibilidad y adaptabilidad al modelo. Existen tres clases de AS en el modelo:

- El agente CSA *Cliente Servidor Agregado* (*CSA*) es el servidor dedicado a realizar las actividades de la aplicación. Ejemplos de éstas funciones específicas son: la emulación del comportamiento de algunos animales, la coordinación de un conjunto de información educacional, un servicio de verificación de correo, etc. Se trata de aplicaciones generadas de forma independiente de quienes creamos GeDA-3D, en distinto tiempo y tal vez de distintas líneas de investigación.

- El agente *Cliente Agregado* (*CA*) trabaja bajo el control del CM. Cuando un usuario ordena la activación de un servicio, El CM valida el proceso mediante consultas a la BD de información del usuario y de los servicios. Si el acceso es concedido entonces el cliente maestro suspende la ejecución local del ambiente virtual y abre paso a la ejecución

[1]La seguridad del sistema, actualmente, está implementada con CORBA. Sin embargo, las condiciones para implementar un agente de seguridad están dadas, a fin de que se realicen tareas de autetificación. En nuestro caso éstas tareas no forman parte del objetivo de nuestra tesis.

del cliente agregado en el mismo espacio de la interfaz. Cuando la aplicación haya terminado, el cliente agregado cede de nuevo el control al cliente maestro y se restaura el ambiente tridimensional.

- El agente *Cliente Administrador* (*CAdmin*), tiene la misma estructura que el CM, pero sus competencias son las de consulta y modificación de las BD. Es decir, puede obtener reportes del listado de usuarios o de los servicios inscritos; además, puede modificar los campos de identificación, claves de acceso, etc... Crear un cliente especial independientemente hace al resto de los clientes mas ligeros, ya que no existe la necesidad de que implementar en cada uno de ellos funciones de manejo de información.

3.3.2 Una Sesión en GeDA-3D

Para que un usuario utilice el sistema GeDA-3D, debe tener instalado previamente el agente CM del sistema. Este agente establece contacto con el agente SM con el fin de establecer la conexión, para recibir el ambiente virtual y tener acceso a los servicios de la aplicación. Al iniciar la ejecución de una aplicación, el sistema requiere algunos datos para crear una configuración o perfil con la que el SM identificará al usuario. La información requerida son la identificación del usuario en el sistema, los datos que usará el agente CM para encontrar al agente SM, su dirección de internet (IP) numérica o alfanumérica legales y el puerto en que las peticiones del agente (usuario) serán atendidas.

Una vez que los datos han sido recopilados, y han sido declarados válidos (la validez se realiza cotejando los datos proporcionados con los almacenados en las bases de datos), los agentes CM y SM comienzan a intercambiar información. Si el SM concede el acceso al usuario, entonces éste dispondrá de la interfaz tridimensional y del servicio de charla hasta su desconexión.

Al principio se transmite enteramente el ambiente tridimensional y

en las siguientes sesiones, esto pasa solamente cuando se detecta alguna modificación en ella. La forma y estilo de la interfaz es definida por quien ha ejecutado el sistema, particularmente, al activar el agente SM.

La comunicación informal tiene un espacio especial en la aplicación del usuario, ahí pueden verse a todos los usuarios conectados y se puede establecer una conversación con ellos. Todas estas capacidades, son realizadas por los agentes CM y SM.

En este punto, el usuario tiene asignado un avatar como su representación en la interfaz 3D, puede navegar en el ambiente, observar a los demás usuarios y ser observado por otros. Ya en este nivel, se pueden realizar una serie de tareas que requieren comunicación directa entre un grupo de personas.

Hay en el ambiente algunas figuras especiales, a las que llamamos sensores. Cuando un usuario se aproxima suficientemente a un sensor e indica que desea activar esta figura, el sensor permite que de desencadenen algunas acciones para proporcionarle un servicio. Este mecanismo hace que el agente CM suspenda su actividad despues de dar aviso al agente CA de que es el momento para intercambiar información con el agente CSA y que de esta forma muestren al usuario una aplicación en particular dentro del ambiente virtual.

Mientras esto sucede, el usuario observa la activación de la figura, se guarda el estado de su interfaz, sus posiciones y vistas, y se inicia una animación desde la figura referida que indica el inicio de la aplicación. Estas aplicaciones son los agentes CSA, que como dijimos, son implementados por distintas personas y es software de naturaleza variable. Que puede ir desde juegos, correo, la representación de un animal y su comportamiento, un programa para la creación de diagramas de ingeniería de software en conjunto, comercio electrónico, un pizarrón distribuido, software relativo a la transmisión de voz o video, etc. implementando incluso sus propios mecanismos de autenticación. Durante todo este proceso, el usuario puede seguir conversando con el módulo de charla ("chat") con el resto de los

usuarios.

Cuando el usuario decida terminar la aplicación que se abrió, por haber activar el sensor, entonces inicia una segunda animación, con el fin de proporcionarle al usuario, amigablemente, la vista del ambiente virtual en el estado en que fue suspendido por el agente CSA. Entonces se rompe la comunicación entre los agentes CSA y CA, posteriormente se rompe la existente entre los agentes CM y CA, y finalmente se reinicia la relación entre el CM y el SM. De esta manera, el usuario utiliza básicamente el sistema GeDA-3D.

3.3.3 Funcionamiento de la comunidad de agentes

En esta sección describimos el funcionamiento de cada uno de los agentes que integran la comunidad de nuestra arquitectura GeDA-3D.

El agente *Servidor Maestro o SM*

El agente SM permite la integración de todos los servidores dedicados o agentes CSA, mediante su registro como clientes del sistema. Además, permite la integración de los usuarios almacenando su información para verificación y consulta. Para ello controla dos Bases de Datos (BD). La BD de usuarios contiene los identificadores, nombres reales, claves de acceso, direcciones de correo electrónico y un campo de información variable para adecuarse al tipo de aplicación implementada. Mientras que la BD de servicios contiene sus identificadores, su estado (disponible o no), dirección IP y número de puerto donde el agente CSA atenderá a los respectivos agentes CA con las peticiones de los usuarios.

Por otro lado, en el SM, también se mantiene el estado de la interfaz. Se ha diseñado una estructura de datos dinámica para almacenar las relaciones de la posición de los usuarios y de los servicios, lo que constituye la base del mecanismo del multicast implementado para distribuir las actualizaciones del

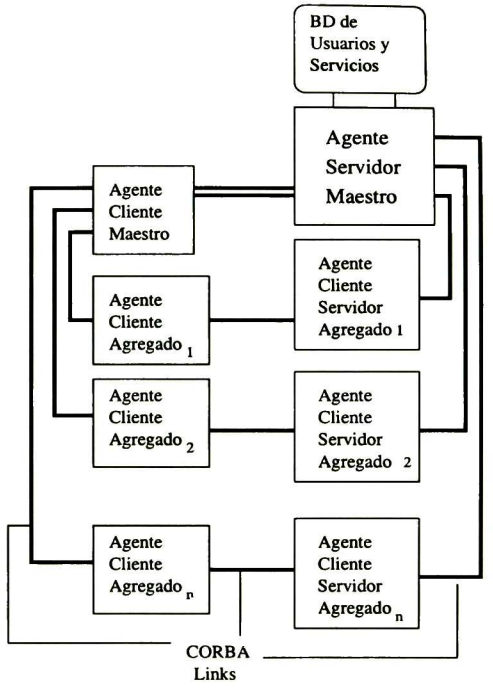

Figura 3.3: Arquitectura Genérica Virtual Distribuida GeDA-3D

escenario hacia todos los clientes. El buen funcionamiento de la arquitectura se basa completamente en la información de ésta estructura de datos y la que reside en las BD. En el diagrama de la Figura 3.3 observamos la propuesta de nuestro proyecto GeDA-3D.

Como mencionamos, en el ambiente el usuario es capaz de navegar y manipular algunos objetos, además puede reconocer a los otros miembros del sistema e interactuar con ellos.

Agente Cliente Servidor Agregado o CSA

El uso de una aplicación que sigue nuestra arquitectura, depende ciertamente de los servicios que se ofrecen. Cuando un usuario desee desarrollar e inscribir un agente CSA, debe cumplir con algunos requisitos relativos a la estructura

de este software. Durante la implementación de estos componentes creamos un código base o plantillas[2] que contienen dos estructuras o esqueletos con las características suficientes para poder comunicarse con el agente SM. Las plantillas creadas son una para el agente CSA y otra para el agente CA. Si se sigue la documentación adecuadamente, la implementación de un servicio resultará sencilla.

Agente Cliente Agregado o CA

El agente CA es pasivo hasta que el agente CM le indica que es su turno en la ejecución del sistema. Mientras los agentes CA y CSA no han sido implementados, es decir, solamente cuentan con el esqueleto básico para su comportamiento, tienen funciones limitadas. Es decir, al iniciar el agente CSA se inscribe con el SM como servicio disponible y es capaz de aceptar conexiones de un agente CA. El agente CA puede conectarse y desconectarse del anterior sin obstáculo alguno. Como podemos deducir, la mayor parte de las funciones de estos agentes dependen de la decisión de los desarrolladores del servicio.

Agente CAdmin

El agente CAdmin, como describimos en la sección 3.3.1, proporciona un acceso de dominio total (de uso y abuso) al sistema GeDA-3D. Decidir la existencia de este cliente no fue tarea sencilla. Al principio, el enfoque de este trabajo pretendía que todas estas actividades se controlaran por sí mismas, esto es, la intervención humana no sería siquiera mínimamente esencial. A través de un agente denominado supervisor, se harían depuraciones en la BD y se restringirían los accesos a usuarios y agentes CSA. Sin embargo, estudiamos y comparamos este aspecto en contra de otros sistemas, como LINUX, en donde existe un administrador general (root) y en Windows NT

[2]Véase el Apéndice A de la Utilización de la Arquitectura, en donde se especifican todos los elementos de la arquitectura, así como las plantillas.

también debe existir un usuario con mayores privilegios para el control de cierta información. La conclusión fue que existen algunas funciones, en donde necesariamente está involucrada la decisión subjetiva de un hombre.

3.4 Metodología de desarrollo

Durante el desarrollo de este trabajo, consideramos algunas técnicas de ingeniería de software (OMT[68], Fusion[69], Jourdon[70]) y algunas herramientas de desarrollo[3] como Platinum[71], Rational Rose[72], que, en general, manejábamos al inicio de este trabajo. De las técnicas analizadas entre las cuales están las descritas por Pressman[75], unidas a nuestra experiencia como desarrolladores, seleccionamos una metodología de desarrollo propuesta por Boehm[74]. Esta selección se hizo con base en un análisis sobre las conveniencias de utilizar esta técnica, ente las cuales están: la mejora ciertas características del ciclo de vida clásico del software y la creación de prototipos, elemento importante de análisis de riesgo.

La metodología de Boehm se basa en las siguientes cuatro actividades:

Planificación. Determinación de objetivos, alternativas y restricciones.

Análisis *de riesgo.* consiste de un análisis de alternativas e identificación/resolución de riesgos; prueba de herramientas y estimación de tiempos.

Ingeniería. Desarrollo del producto de un nuevo nivel.

Evaluación *de los resultados.* Valoración de los resultados de la ingeniería.

Un aspecto intrigante del modelo en espiral (Figura 3.4) se hace evidente cuando consideramos la magnitud de su dimensión radial, varios autores consideran que esta magnitud es controlable solamente por las habilidades

[3]Llamadas Herramientas CASE o lenguajes de 5ta Generación.

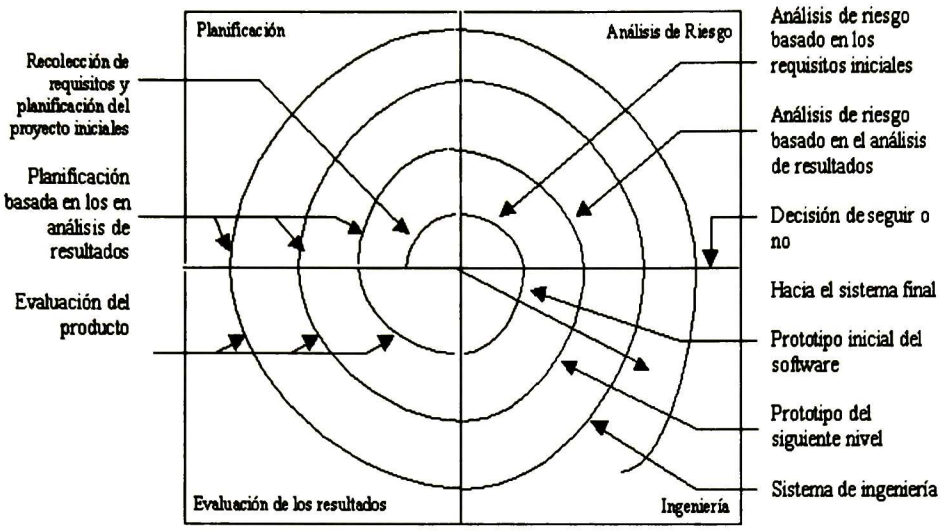

Figura 3.4: Modelo en espiral

de planificación de los desarrolladores. Esto es, si se hace una inadecuada planificación, entonces la espiral puede crecer indefinidamente sin alcanzar el éxito en el desarrollo de un proyecto.

Con cada iteración alrededor de la espiral (comenzando en el centro hacia el exterior) se construyen prototipos sucesivos del software, cada vez más completos.

Durante la primera vuelta alrededor de la espiral se definen los objetivos, las alternativas y las restricciones, y se analizan e identifican los riesgos. Si el análisis de riesgo[75](capítulo 4) indica que hay una incertidumbre en los requerimientos, se puede usar la creación de prototipos en el cuadrante de ingeniería para mejorar su definición usando simulaciones y otros modelos a fin de definir mas el problema y refinar este requisito. El paradigma del modelo en espiral para la ingeniería del software es, en opinión de Pressman y nosotros estamos de acuerdo, el mas realista para el desarrollo de software y de sistemas a gran escala. Utiliza el enfoque evolutivo descrito por Gilb[?], permitiendo al desarrollador reaccionar a los riesgos en cada nivel evolutivo.

Utiliza la creación de prototipos como un mecanismo de reducción del riesgo, pero, lo que es más importante, permite a quien lo crea aplicar el enfoque de utilización de prototipos en cualquier etapa de la evolución del producto. El aspecto mas crítico de este modelo es el hecho de que no siempre se puede estar seguro de que el enfoque evolutivo es controlable. Requiere una considerable habilidad para la valoración del riesgo, y cuenta con esta habilidad para el éxito.

El modelo original presentado por Pressman[75] fue ligeramente modificado para nuestros fines. En general, la presentación del producto en los refinamientos sucesivos se somete a la aprobación del cliente, y en nuestro caso esta fase fue dedicada al análisis de los resultados y reajuste de los requerimientos. para el inicio de la implementación de la siguiente fase. El desarrollo de la arquitectura y del prototipo se realizaron en 3 fases:

1. *Comunicación.* En esta etapa se desarrolló la parte de la arquitectura encargada de mantener el canal de comunicación entre el agente CG y los agentes de interacción.

2. *Integración.* Al terminar esta fase, se tuvo como resultado la integración entre lo obtenido en la fase anterior, el medio de comunicación entre CG y agentes de control, las funciones de registro y control de usuarios y algunas funciones de coordinación.

3. *Servicios.* En esta fase se definen varios aspectos, tales como la caracterización de un caso de prueba con lo que se tienen elementos suficientes para implementar los agentes de control. Al implementar esta etapa se contempló la integración de los agentes de control al ambiente y las modificaciones pertinentes para la interacción del servicio CM con el agente CG, además del intercambio de información entre CG y los agentes de interacción o nodos finales.

3.5 Especificación de GeDA-3D

Para realizar la especificación se analizaron varias metodologías, entre ellas, LOTOS[78], la aplicación de técnicas que involucraban el uso de la lógica temporal y espacial[77] y el modelo de arquitectura de software propuesto por Hofmeister, Nord y Sodi[79]. Después de evaluar los siguientes factores:

- Nuestro objetivo: Especificar claramente el comportamiento de nuestro sistema.
- El tipo de aplicación que desarrollamos
- El tipo de sistema que es nuestro caso de prueba[4]
- Las capacidades de estas metodologías para representar un sistema

Coincidimos con los autores Hofmeister, Nord y Sodi[79] en que una buena especificación de la arquitectura construye el éxito de un producto. Y en que esta especificación se basa en que el diseño de la arquitectura debe corresponder con los requerimientos definidos entre los desarrolladores. Para definir el comportamiento de nuestra arquitectura GeDA-3D hemos decidido usar el lenguaje formal de especificación LOTOS.

3.5.1 Descripción de la metodología

Desde 1980, cuando un gran conjunto de investigadores trataban de modelar el comportamiento de múltiples sistemas, surgió LOTOS(del inglés, Language Of Temporing Ordering Specification) como un lenguaje de especificación formal, basado en el álgebra de procesos y en el ordenamiento secuencial de las acciones. En particular, utilizaremos Basic Lotos que es capaz de caracterizar la sincronización de procesos de un sistema dado. Una especificación está formada por un conjunto de expresiones de

[4]Cuya definición detallada se verá en el capítulo 4

comportamiento. Estas expresiones contienen son procesos que se sincronizan a través de un conjunto de eventos. Las expresiones de comportamiento están regidas por un conjunto de reglas de derivación o de producción que permiten inferir el comportamiento y validar o verificar la especificación.

3.5.2 Aplicación de la metodología

Para llegar a los detalles de la especificación iniciaremos con plantear en términos llanos el algoritmo que muestra el comportamiento general del sistema. Posteriormente, mostraremos el arbol de sincronización que determina el comportamiento del caso de uso que estamos contemplando, para terminar con la especificación formal respetando la sintáxis de LOTOS.

Algoritmo de Comportamiento General

A continuación enumeramos los pasos que explican la conducta de GeDA-3D. Entendamos que estos pasos sólo muestran las funciones de comunicación, esto es, actividades que tienen que ver con navegación en la interfaz o procesamiento de datos especial no se contemplan.

1. Los datos del usuario son proporcionados al agente CM
2. El agente CM envía los datos al agente SM para su verificación
3. Si la verificación del paso 2 muestra que el usuario es bienvenido al sistema, entonces pueden suceder distintas acciones:
4. CM espera la señal de activación de un servicio desde el exterior del sistema
5. Al presentarse la señal de activación:
6. CM se conecta con CA
7. CM se suspende y espera la señal de terminación de CA

8. CA se conecta con CSA y espera la señal de salida desde el exterior del sistema

9. Al presentarse la señal de salida se activa la señal de terminación

10. Con la señal de terminación CM se reactiva lara volver al paso 4

11. Si la verificación del paso 2 muestra que el usuario no es bienvenido al sistema, entonces volver al paso 1

Formalización del comportamiento

La siguiente especificación de comportamiento concentra las funciones del caso general de uso del sistema. Presentaremos el árbol de transición o sincronización y, después de la expresión de comportamiento correspondiente, la especificación en Basic LOTOS probada con la herramienta ARA.

En la siguiente Figura(3.5) vemos el árbol de sincronización o de transición obtenido del comportamiento general de la Arquitectura GeDA-3D. Considerando solamente, como se advierte en la bibliografía[78] correspondiente, los eventos que desencadenan a las funciones de comunicación entre los agentes participantes.

La expresión de comportamiento que cumple con el algoritmo antes presentado es la siquiente:

datos;(((rechazo;(datos[]EXIT)[](valida;((((navega;(actserv[]EXIT))[](acepta;actserv;pca csa;(EXIT[](break;navega))))[]EXIT)))[]EXIT)

El significado de los eventos o compuertas que se muestran en la figura, es el siguiente:

datos	El usuario introduce los datos necesarios para ser validado y establecer una conexión con entre los agentes cm y sm
rechazo	El sistema no autoriza el acceso del usuario

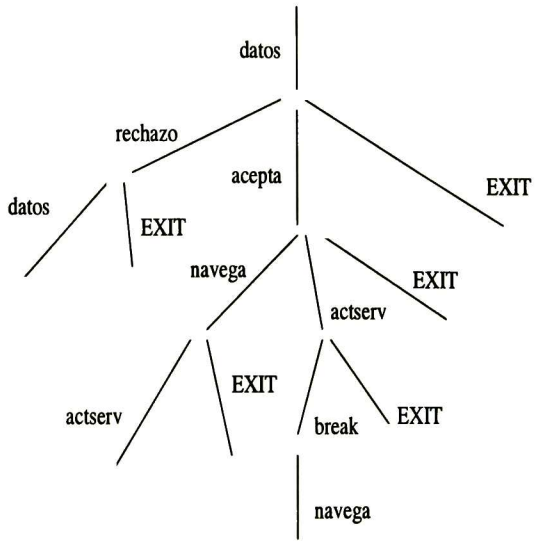

Figura 3.5: Arbol de transiciones para el comportamiento de GeDA-3D

valida — El usuario solicita que los datos que ha proporcionado sean cotejados para su validación

navega — El usuario navega en el ambiente virtual

actserv — El usuario desea activar un servicio del ambiente

acepta — El sistema autoriza el acceso del usuario

pca-csa — Proceso de comunicación entre los agentes cm y ca, además de ca y csa

break — Proceso indicador de que el proceso pca-csa ha terminado

EXIT — El usuario sale del sistema

A continuación, escribimos la descripción formal en Basic LOTOS que ha sido probada con las herramientas ARA:

```
process Using_ GeDA3D [datos, respuesta, moves, actserv, stopserv] :=
valida[datos, respuesta]
|[respuesta]|
(
        (respuesta;(*El evento se consuma*)
        accion[stopserv, moves, actserv]
        |[moves, actserv]|
        (
                (
                        ñavega[moves, actserv]
                        |[actserv]|
                        (accion[actserv, moves, stopserv] [] stop)
                )
                []
                (
                        pcacsa[actserv, stopserv]
                        |[stopserv]|
                        accion[stopserv, moves, actserv]
                )
        )
        []
        EXIT
        )
)
where
        process valida[info]:noexit:=
                info;
        endproc

        process accion[mov, act, stopserv]:=
```

```
            (mov; stop)
            []
            (act; stop)
            []
            stop
        endproc

        process navega[mov, act] :=
            (mov; stop)
            []
            (act; stop)
        endproc

        process pcacsa[act, stopserv] :=
            (act; stop)
            []
            stop
        endproc
    endproc
```

De esta forma, LOTOS y ARA nos han permitido especificar el comportamiento mas general del sistema, sin embargo, esta técnica podría ser utilizada para describir todos los casos de uso y funciones del sistema, de tal forma que en conjunto se cree una estructura general, a manera de grafo de tareas, que muestre todos los estados en que se puede encontrar un sistema dependiendo de los eventos que se han presentado y a la gama de acciones permitidas despues de cada sucesión de ellos.

3.6 Alcances

La principal característica de GeDA-3D es que para su concepción no se considero ninguna hipótesis con respecto a algún tipo de aplicación especifica. Esto se traduce en una gran libertad de uso para los desarrolladores. El género de los sistemas que pueden montarse es tan variable como se quiera, por ejemplo:

1. Relativos a alguna ciencia (exactas, salud, administrativas, etc.).
2. Entretenimiento.
3. Educación.
4. Investigación.
5. Diseño.

No obstante, esta categorización contiene sólo un subconjunto de las clases de aplicaciones que pueden adaptarse al proyecto.

Alguna vez, hicimos una fuerte comparación con el potencial del lenguaje HTML y los navegadores como Netscape, Internet Explorer y HotJava.

Así como cualquier persona puede crear una página en formato html, podrá programar en cualquier lenguaje una aplicación en nuestro sistema. Siempre y cuando como se respeta la estructura html, se respete la estructura de las plantillas o esqueletos de los agentes CSA y CA.

Así como los navegadores pueden interpretar este formato, nuestro agente SM podrá comunicarse con los agentes CSA y CA correctamente modificados.

La internet esta construida con base en millones de servidores alrededor del mundo con incontables tipos de información; así mismo, nuestra Arquitectura Genérica Virtual Distribuida GeDA-3D tiene el potencial para convertirse en una gran red de agentes con innumerables géneros para los ambientes virtuales y servicios a ofrecer.

3.7 Conclusión

En este capítulo, hemos descrito GeDA-3D, una arquitectura robusta y confiable para el desarrollo de sistemas distribuidos cooperativos. La principal característica de esta arquitectura al que se pueden integrar un conjunto de aplicaciones de naturaleza variable. Además, esta arquitectura ofrece la facilidad de manejar una interfaz 3D de manera sencilla. Para validar nuestra propuesta, especificamos formalmente la relación entre dos de los agentes más importantes del sistema.

Capítulo 4

Aplicación

4.1 Objetivo

En este capítulo mostraremos un caso de prueba para demostrar la funcionalidad de nuestra arquitectura genérica GeDA-3D. Este prototipo es un sistema llamado Oficina Virtual (OV).

4.2 Introducción

La evolución de los medios de comunicación en los últimos años, ha llevado al ser humano a confiar en estos medios de nueva tecnología para desarrollar sus actividades cotidianas. Dado que Internet, constituye el medio de comunicación mas prolífico actualmente, nosotros la utilizamos para mostrar la aplicabilidad de nuestra investigación y producto generado. Es decir, decidimos implementar una Oficina Virtual basada en GeDA-3D debido a que es un problema y aplicación familiar para gran parte de los usuarios y porque es un caso de estudio muy completo del CSCW[57][58][60][61].

Es necesario aclarar que, nuestra aplicación de Oficina virtual ha sido objeto de estudio y que existen algunos resultados anteriores al inicio de esta investigación. En específico, se realizaron en el Centro de Investigación y

Estudios Avanzados del Instituto Politécnico Nacional, dos prototipos de un sistema distribuido cuya interfaz era también tridimensional. Sin embargo, estos proyectos dependían de un navegador de internet y pretendían, a través de herramientas comunes al desarrollo de páginas web, integrar una serie de servicios específicos. Estos prototipos eran aplicaciones con una identidad perfectamente definida, una oficina virtual. Y cerrados, en el sentido de que sólo se podían extender conociendo la estructura total del sistema a fin de modificar el programa adecuadamente.

Nuestro prototipo retoma algunas de las ideas generadas en el seno de nuestro equipo de trabajo, pero la idea es renovar en absoluto el enfoque de este tipo de aplicaciones. Así, con esta mentalidad, construimos sobre GeDA-3D un ambiente virtual con apariencia de una oficina virtual, personajes para cada usuario que participan dentro de esa oficina, habitaciones y objetos que representan sensores asociados a un conjunto de servicios estándar, por ejemplo, la consciencia de un solo horario registrado por el agente SM. Designamos a los servicios como estándar porque estos servicios son los que generalmente se ofrecen o bien, se encuentran en una oficina real.

A continuación describiremos, en general, la estructura del sistema, después la división de las tareas en el grupo de desarrollo de esta tesis para identificar el aporte al proyecto de esta investigación particular enumerando las pruebas hechas al sistema y herramientas evaluadas durantela implementación correspondiente.

4.3 Estructura del Sistema

Estrictamente hablando, el núcleo de la OV es GeDA-3D. Lo que significa que los agentes descritos: SM, CM, CA, CSA y CAdmin efectúan en este plano las funciones anteriormente descritas. No obstante, como es imaginable y de acuerdo a lo tratado en el capítulo anterior, las principales diferencias existen en la definición de los agentes CSA y CA.

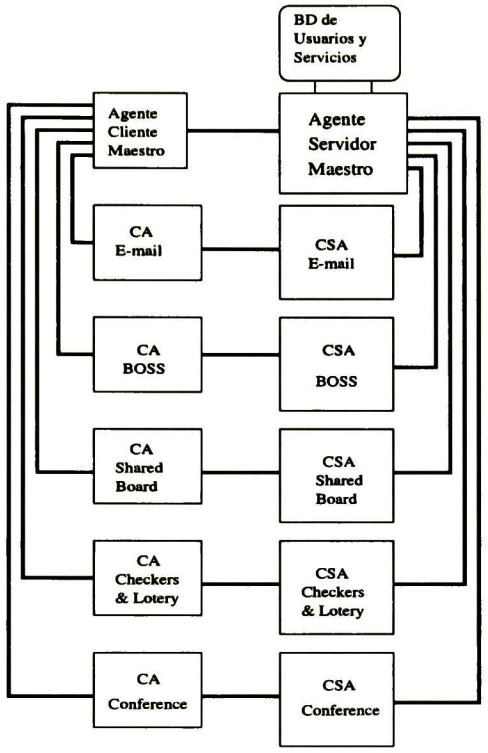

Figura 4.1: Arquitectura de la Oficina Virtual, basado en GeDA-3D.

En esta sección se enumerarán y explicarán las funciones específicas de los agentes CSA. Es decir, escribiremos el comportamiento de los servicios ofrecidos en la OV En la Figura 4.1 se muestra cómo la estructura de la OV tiene como centro y base a GeDA-3D. Esta misma visión tendrá toda aplicación basada en nuestra arquitectura GeDA-3D.

En la Figura podemos ver que las extensiones a la arquitectura GeDA-3D son agentes CSA. En la OV, los CSA realizan las siguientes tareas: Agenda de Citas y Correo, ejecutado por el agente Bussines Organizer Schedule System (BOSS); Conferencias, llevado a la implementación por el agente Conference; Pizarrón, con el agente SharedBoard; Juegos, que es ejecutado por dos entidades, el agente Checkers y el agente Lotery

Antes de pasar a la descripción, debemos resaltar que uno de los aspectos más importantes en estos sistemas es la comunicación, así que tal como otros investigadores[81] creímos necesario que los servicios ofrecidos faciliten distintos tipos de intercambio de información. Es decir, de alguna manera, los agentes que describiremos a continuación tienen actividades que involucran la comunicación, en modo síncrono o asíncrono. También es muy importante que quede claro que cada servicio o tarea en la lista es realizado por dos partes: el agente CSA que mantiene el servicio y el CA que es la herramienta del usuario para utilizar el servicio.

4.3.1 Servicio BOSS

Este servicio coordina el envío y recepción de mensajes Internos y Control de Citas. Ambas actividades aparecerán en el mismo espacio indicando quien lo envía y quien lo recibe, el contenido del mensaje y si se trata de un mensaje o si es una cita, en el segundo caso, se debe indicar si se trata de una cita general o si se llevará a cabo en una habitación reservada para que el agente Conference realice su función. Por lo que, aun que no hay interacción entre los distintos agentes CSA se considera una reunión de alta prioridad.

Mensajes Internos

Los mensajes internos se refieren al intercambio de texto entre los usuarios inscritos al sistema. En general, este agente CSA tendrá un espacio para guardar la información, mensajes internos de cada usuario, de tal manera, que al iniciar el agente CA se hacen las consultas o modificaciones solicitadas.

Al utilizar este servicio, el usuario recibe una notificación indicando si tiene o no mensajes nuevos, y en el caso afirmativo comienza a recibirlos. En el espacio del servicio se proporcionan herramientas para enviar mensajes, y los formatos a llenar son tal y como se han estandarizado en la mayoría de los programas comerciales. Si el destinatario se encuentra se le notifica

inmediatamente, de lo contrario, se le avisará de este nuevo mensaje cuando ingrese al servicio.

Citas

Este servicio permite establecer una fecha, hora y lugar entre un grupo de personas que se reunirán en algún punto de la OV. Este servicio puede interactuar con otros, por ejemplo con el servicio de Conferencias, para restringir la lista de asistentes y asegurar que todos ellos han recibido el aviso de la reunión.

El organizador y su cargo le dan cierta prioridad a la cita, por ejemplo, si se trata de un usuario con cargo gerencial entonces ésta se considera de carácter urgente y el horario no puede ser desplazado por ninguna razón. La cita contiene la lista de asistentes, el lugar, el día, la hora y el asunto a tratar. Si el lugar es la habitación llamada sala de conferencias entonces este agente CSA debe considerar una reunión de mayor importancia, por lo que se ofrecen al usuario otras capacidades par hacer consideraciones y restricciones.

Este servicio también verifica que el horario fijado para la cita esté libre en el tiempo de todos los usuarios y no determina una cita ni la notifica a los invitados hasta que eso se cumpla. De igual forma que con los mensajes, si los invitados están conectados al sistema, se les notificará inmediatamente.

Correo

En esta faceta del servicio se facilita el envío y recepción de mensajes vía electrónica a través del sistema, además se pueden enviar archivos adjuntos (attachment). La implementación de este servicio involucra la investigación del funcionamiento de los protocolos POP3 y SMTP. Las funciones que se pueden utilizar con este servicio son Revisión, Composición, Envío y Reenvío de mensajes. Cada una de estas funciones se maneja de manera estándar, para explotar las capacidades del usuario al manejar el sistema.

4.3.2 Servicio de Conferencias

El organizador de una cita cuyo lugar fue establecido en la sala de conferencias debe registrar la reservación de la habitación. En donde se pueden llevar a cabo exposiciones con material gráfico o bien, reproducción de filminas. Como se mencionó en la Introducción, un elemento básico de este sistema es la consciencia de un solo tiempo para todos los participantes. Ya que esto determina la diferencia entre las actividades que se realizan en modos síncronos o asíncronos.

Se reservará la sala y las herramientas de la exposición por tiempos específicos y para listas de asistentes estrictamente definidas. Minutos antes del inicio de cada conferencia se notificará a los invitados para asegurar su participación.

4.3.3 Servicio de Pizarrón

En este servicio se proporciona un área de trabajo común, para que los usuarios realicen presentaciones con las herramientas de texto y dibujo ofrecidas:

a)Puntos
b)Mano alzada
c)Líneas rectas y curvas
d)Polígonos regulares e irregulares
e)Círculos y ovalos
f)Inundación
g)Texto y fuentes
h)Borrado
i)Selección de área y color
j)Aerógrafo
k)Zoom

El espacio total de dibujo es subdividido de acuerdo al numero máximo

de usuarios en este pizarrón, incluso se pueden compartir algunas áreas o enrocar las asignaciones para facilitar la participación conjunta.

4.3.4 Servicio de Juegos

En este servicio se proporcionara la activación de dos juegos:

- Damas inglesas (Agente Checkers)
- Lotería (Agente Lotery)

El comportamiento de estos agentes es propio a las reglas del juego, por lo que no lo describiremos aquí. Sin embargo hay que resaltar que se trata de juegos en donde participan grupos de dos o mas personas y que la sincronía es un factor de cuidado para que estas actividades funcionen bien.

4.4 Desarrollo Compartido

Como se mencionó en la metodología de desarrollo de la arquitectura, la implementación de la arquitectura y del prototipo de la Oficina Virtual se realizó en tres fases: Comunicación, Integración y Servicios.

Dado que esta tesis ha sido realizada en conjunto con la Licenciada María Eugenia Puga Nathal, es necesario explicitar lo siguiente con respecto al desarrollo del presente trabajo: La investigación documental, así como el diseño de la arquitectura presentada fueron resultados conjuntos; como podemos observar en el diagrama de actividades de la Figura 4.2, la división del trabajo fue natural, la divergencia en la implementación creó además, divergencia en nuestras investigaciones por lo que:El desarrollo que presentamos a continuación contempla: El desarrollo del Agente SM y las plantillas para los agentes CM, CA, CSA y CAdmin. Atendiendo, en el primer caso, a las funciones de manejo de las BD, seguridad del sistema, mantenimiento del estado global del ambiente virtual y control de los agentes

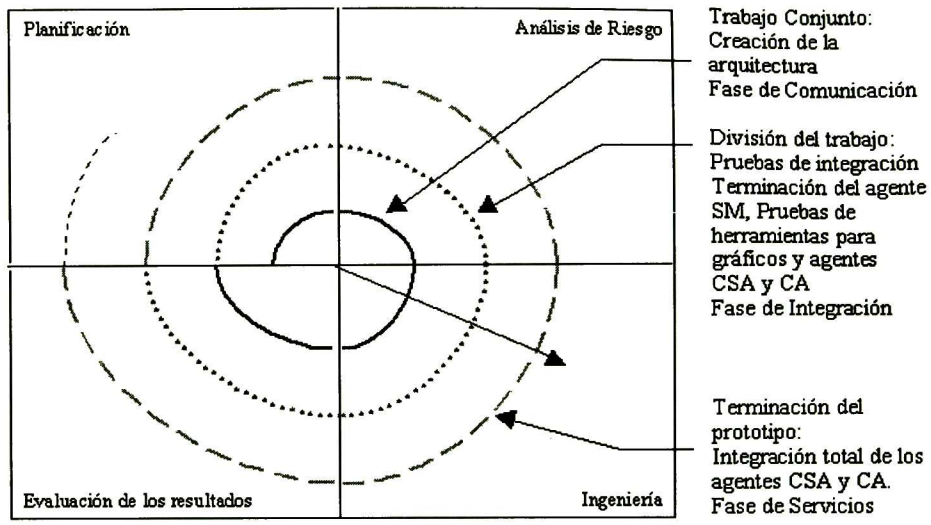

Figura 4.2: Diagrama de actividades en el desarrollo de esta tesis

CSA. La investigación realizada se puede leer en las pruebas de herramientas, para la implementación de la arquitectura GeDA-3D y de la Oficina Virtual de la sección 4.5.

Por lo tanto en el reporte de tesis de maestría de mi colega María Eugenía Puga, abarca parte de la investigación sobre GeDA-3D y el desarrollo de los servicios expuestos en nuestro caso de prueba. Existe un fuerte trabajo de investigación e implantación de muchas herramientas para llevar a buen término los múltiples servicios que ya describimos.

4.5 Implementación

Inicialmente, era claro que utilizaríamos el lenguaje de programación Java de Sun MicroSystems (TM) por su característica en la generación de programas multiplataforma. Para describir las herramientas probadas y utilizadas haremos referencia a las etapas de desarrollo ya mencionadas.

Comunicación

Por nuestros objetivos, sabíamos que requeríamos de tecnologías para sistemas distribuidos y de invocación remota. Existen diferentes tecnologías entre las cuales encontramos RMI[82], DCOM[21] y CORBA[19]. Nuestra decisión se inclino hacia corba, ya que en ese momento no estaba atada a ningún sistema operativo ni lenguaje como ocurría con las otras opciones. Así, decidimos utilizar CORBA como plataforma de comunicación. Este estándar de la OMG permite a los desarrolladores de agentes servicio utilizar cualquier lengaje en sus desarrollos y provee una interoperabilidad de hardware y de sistemas operativos.

CORBA nos proporcionó transparencia en las funciones de comunicación entre los componentes. Recordemos que para un objeto de CORBA, los objetos remotos tienen apariencia de objetos locales, debido a que el proceso de comunicación entre ellos no resulta diferente en estos casos. Además, cualquier objeto puede hacer una invocación remota con respecto a otro, siempre que tenga una referencia adecuada de él.

Haciendo uso de las facilidades de CORBA, inscribimos a los agentes CA y a los agentes CSA, como objetos de CORBA, lo cual les permite comunicarse entre sí y con el agente SM. Esta es la manera que implementamos la comunicación entre todos los elementos del sistema.

La referencia de un objeto remoto puede entenderse si nos auxiliamos del concepto de apuntador en programación, ya que nos permite accesar directamente a los atributos y métodos de un objeto. Ésta es calculada por los módulos de CORBA lo que nos permitió tener independencia entre nuestro proyecto y los detalles de la conexión entre los objetos. Debido a que las referencias entre los objetos son unidireccionales, esto es, solo un objeto puede invocar a otro mediante una relación, en la Figura 4.3 ilustramos el sentido en que fluye la comunicación entre los componentes de la arquitectura, considerando el modelo de invocación remota de CORBA. En el caso del agente CSA definimos la extensión de un canal de comunicación,

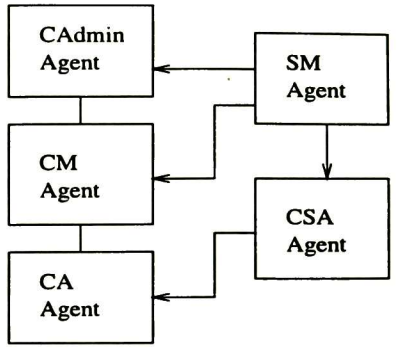

Figura 4.3: Sentido de la comunicación en la invocación remota con CORBA (explicando lo que significa el sentido de las flechas)

esto es en forma de dos relaciones entre los tres objetos SM CSA CA.Una vez que se decidió por CORBA para implantar la comunicación, se realizo una investigación sobre las implantaciones existentes de CORBA. Esta investigación nos llevo a que existen diferentes implantaciones del corazón de CORBA que es el ORB, entre las que investigamos haciendo pruebas están la de Sun MicroSystems en el lenguaje de programación Java (jdk versión 1.2) y el ORB distribuido por la OMG llamado ORBacus u Omnibroker en su versión 3.2.

En la Figura 4.4 se observan los resultados de las pruebas que realizamos para decidir de entre ellos cual utilizar en nuestra implementación. Estas pruebas consistieron en la evaluación de dos programas, uno, de la impresión simple de un mensaje, sin invocación remota que obviamente produjo resultados semejantes para los ORBs. En la tabla, se ven los valores en milisegundos correspondientes a las columnas debajo del título "Println local" El segundo programa, funcionaba entre dos objetos; en donde uno hacia la invocación remota de un método que se implementaba en el otro. Como podemos ver, los resultados favorecieron al ORBacus con el 50% menos de tiempo sobre el ORB integrado al Java. Los números que reflejan lo dicho en los números de las columnas llamadas "Enviomsgremoto" Así, se decidió

ORBACUS				ORB de JDK 1.2			
Println local		Enviomsg remoto		Println local		Enviomsg remoto	
2019	1540	4950	4560	1920	1590	8290	7800
1540	1590	4620	4620	1540	1600	7750	7750
1540	1540	4610	4670	1590	2090	7800	7800
1600	1590	1560	4670	2030	1540	7800	7850
1590	1600	4560	4560	1600	1980	7800	7910

Figura 4.4: Tabla de resultados entre los ORBs probados

utilizar al ORBacus para la implementación del proceso de comunicación en nuestra arquitectura.

Integración

La fase de integración del proyecto contempla el aspecto de establecer la buena comunicación entre los elementos, creación de las operaciones básica de control en las BD, soportar el mantenimiento de la parte gráfica del sistema, definir la interfaz de trabajo y considerar cómo sucederá la activación de los sensores. Esas figuras que se ven en el ambiente virtual y que activan los agentes CA y posteriormente CSA para proporcionar un servicio. En la fase de comunicación se realizaron pruebas que facilitaron la agregación de los agentes CA y CSA. Sin embargo, en el desarrollo de la interfaz, hubo que analizar decenas de herramientas gráficas a fin de encontrar aquella que nos proporcionara una buena velocidad en el procesamiento gráfico y que permitiera nuestra intervención con el lenguaje de programación Java.

El estándar gráfico sugerido por el trabajo de Shinkuro[55], OpenGL[80], fue muy prometedor, y encontramos un conjunto de programas que se apoyan en este estándar para crear sus ambientes tridimensionales. OpenGL es un estándar de SGI que trabaja de forma cercana al hardware de video utilizado. En general, se trata de un conjunto de librerías gráficas escritas en C. Estas librerías se compilan para un determinada plataforma y para

un determinado hardware y crean una plataforma de comandos ejecutables. Esos otros programas, que necesitan la implementación del estándar para el manejo de sus propios gráficos, llaman a este conjunto de comandos.

Así pues quedó claro que debíamos encontrar una herramienta programable en Java y que, de ser posible, usara directamente los comandos gráficos.

VRML[45], el lenguaje de preferencia para quienes realizan sistemas de RV en el web, está basado en OpenGL. Java3D, también esta sustentado por este estándar. El problema principal con VRML y Java3D fue su lento desempeño en el despliegue de los gráficos. Las herramientas de programación en el lenguaje aludido y la librería de Sun se mostraron complicadas, con escasa documentación y bajo desempeño. Por lo que aunque no se tomó la decisión de abandonar estas herramientas, se optó por seguir buscando alguna que se mostrara más efectiva.

Encontramos primero el concepto de *bindings* que definía aquello que buscábamos: se trata de una relación directa entre los comandos de hardware generados por la compilación de las librerías y un conjunto de funciones de un lenguaje distinto al que se uso para la implementación de la plataforma gráfica. Entonces fue claro que se tratábamos de encontrar los bindings de OpenGL para Java.

Probamos varios tipos de bindings para Java, entre los cuáles tenemos herramientas como JGL, JOGL, YAJOGL, JavaGL y GL4Java. Para los primeros 4 bindings obtuvimos resultados insatisfactorios, derivados de la evaluación de los ejemplos de aplicación teniendo en cuenta criterios como velocidad y "rendering" o dibujo tridimensional de superficies, además porque algunos de ellos mostraban dependencia con respecto al estándar de OpenGL implementado por Microsoft, lo que significa que esto podría sacrificar nuestro carácter multiplataforma tan celosamente cuidado. GL4Java usa cualquier implementación de OpenGL, léase publicadas por de SGI para varias plataformas o MesaGL para unix, entre otras; este binding tiene un

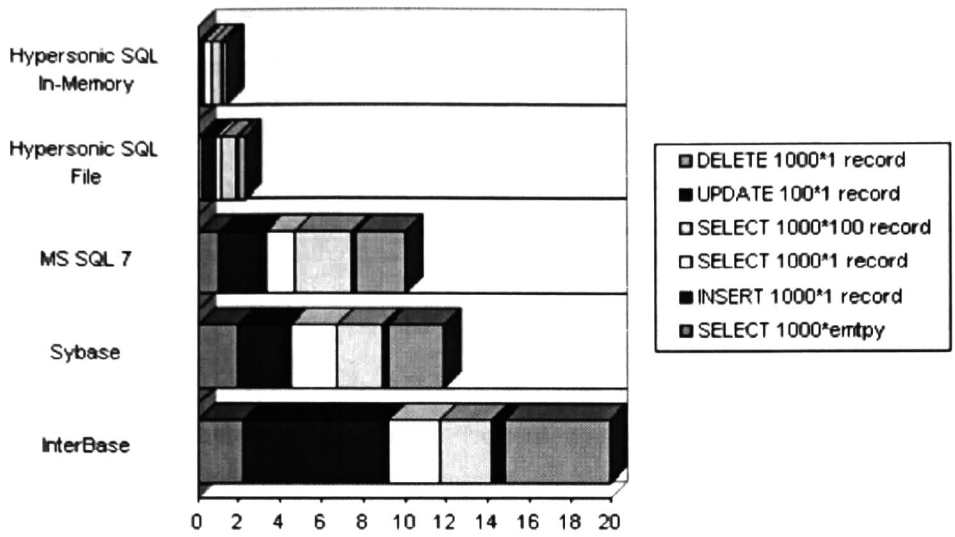

Figura 4.5: Resultados estadísticos en la comparación del rendimiento entre dos ORBs

desempeño notable y está en constante desarrollo. Su autor, Sven Goethel, intercambia constantemente información con todos los usuarios de la librería para mejorar sobre la marcha las características de su producto GNU.

En cuanto a la implantación de las Bases de datos, hubo que iniciar la búsqueda de un manejador adecuado para que interactuara con JDBC, el módulo de control de BD en Java. Los manejadores que probamos fueron Postgress, tinySQL, mySQL y HypersonicSQL. De los cuáles seleccionamos HypersonicSQL debido a que algunos investigadores realizaron pruebas comparando su rendimiento en contra de manejadores comerciales importantes como Microsoft SQL, Sybase e Interbase.

Los resultados de estas pruebas se muestran en la gráfica de las figuras 4.5 y 4.1, con relación a las operaciones de SQL mostradas y el tiempo (en milisegundos) empleado para su ejecución.

Cuando las operaciones de las BD implementadas con HypersonicSQL generan archivos para almacenarse en memoria o en archivo el tiempo

	H-SQL In-Memory	H-SQL File	MS SQL 7	Sybase	InterBase
SELECT 1000*emtpy	0.10	0.24	1.01	1.98	2.25
INSERT 1000*1 record	0.20	0.60	2.29	2.51	7.00
SELECT 1000*1 record	0.41	0.32	1.40	2.22	2.45
SELECT 1000*100 record	0.46	0.69	2.75	2.22	2.47
UPDATE 100*1 record	0.03	0.10	0.23	0.25	0.60
DELETE 1000*1 record	0.19	0.34	2.39	2.69	5.15
Total	1.39	2.29	10.07	11.87	19.92

Tabla 4.1: Tabla de Resultados estadísticos en la comparación del rendimiento entre los ORBs

empleado es considerablemente menor. Este hallazgo nos indicó que este manejador podría auxiliarnos en el desarrollo de esta parte de nuestro prototipo.

Ahora bien, en los inicios de este año, tuvimos en nuestras manos un buen proyecto de investigación terminado, un conjunto de herramientas: Java, GL4Java y HypersonicSQL para comenzar la implementación del caso de prueba de la Oficina Virtual. El resultado de este período de trabajo fue positivo y gratificante.

Capítulo 5

Conclusiones

5.1 Objetivo

En este capítulo presentamos los resultados generados por esta investigación.

5.2 Resultados

- Creamos una arquitectura genérica para Integrar sistemas distribuidos de naturaleza variable
- Proporcionamos en esta arquitectura las herramientas para utilizar un ambiente virtual común a todos los usuarios del proyecto que se implemente
- Creamos los medios para que una aplicación distribuida se adapte a esta arquitectura y obtenga todos los beneficios implementados por los agentes
- Implementamos un prototipo, llamado Oficina Virtual, fundamentada en la arquitectura propuesta. Esto muestra la funcionalidad del

modelo en aplicaciones con dentro del ámbito de trabajo cooperativo y comunicación

- Afirmamos que la implementación de los agentes CSA puede ser independiente y variable, tenemos el gusto de demostrar esto con la integración de la implantación de un servicio, dentro del prototipo de la Oficina Virtual, realizada por la generación 1999-2001 con especialidad en Computación del CINVESTAV, Unidad Guadalajara

5.3 Trabajo Futuro

Nuestro trabajo se ha convertido en la antesala para innumerables aplicaciones distribuidas, por lo que su aplicación no es o no debería ser, en este punto, de valor despreciable.

Sin embargo, las preguntas de ¿cómo mejorar este modelo? y ¿cómo hacer modificaciones a la arquitectura de tal forma que su uso sea aún mas flexible?, sí tienen respuesta.

Actualmente, no existe un medio directo para crear el ambiente virtual de la aplicación. El usuario debe crear un archivo de texto y con la sintaxis especificada, indicar formas, colores y dimensiones para cada objeto del escenario. Por lo que un Editor del ambiente, facilitaría la generación de este archivo a fin de que quien haya decidido instalar la aplicación, o bien el agente SM, encuentre en esto una tarea mas fácil.

Como mencionamos anteriormente, las funciones de seguridad, que son atribuidas al agente SM están implementadas con CORBA, lo que según algunos desarrolladores puede dar lugar a acesos no autorizados, cona respecto a esto, hemos afirmado que hay lugar para un agente de seguridad cuya función sea precisamente evitar estas situaciones a través de técnicas de autenticación o encriptación de claves de acceso para los usuarios y para los agentes con quienes interactúa.

Por otro lado, si el servidor principal tiene una falla importante las

conexiones con los agentes CSA y CA se rompen, lo que da lugar a poca robustez. Este fenómeno puede evitarse con mecanismos llamados "de espejo"(mirror) o de división de la carga de las tareas entre el servidor original, el del agente SM, y otros servidores que conserven la información e interacciones en un mismo estado. Por lo que ante una falla del agente SM, la situación puede ser salvada por alguno denominado agente auxiliarSM.

Bibliografía

[1] ref. CSCW bookstore

[2] Niklaus Wirth, «Algoritmos y Estructuras de datos», Prentice Hall, 1987

[3] Bertrand Meyer,«Object-Oriented Software Construction»,Prentice Hall 1989, 480pp. ISBN 0-13-629049-3

[4] Félix F Ramos C., David R. Garduño B. Iván Romero and José L. Marquez S. «Una Oficina Virtual Distribuida Basada en Agentes», Second International Congress of Electric and Electronic Engineering. Aguascalientes, October 1998, IEEE, México

[5] ref. Definición de Agentes

[6] http://www.omg.org, The OMG web page

[7] Steve Vinoski, «CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments», IEEE

[8] Sulivan, S. O., «The CORBA Services - Enterprise Infrastructure», Proc. of second European Research Seminar on Advances in Distributed Systems (ERSARDS'97)" 1997, March, Switzerland

[9] «ORBACUS 3.1.1.» User Guide, «ORBACUS 3.3.1.» User Guide.

[10] OMG, «Common Object Request Broker Archictecture», July, 1995.

[11] OMG, «Common Object Services Specification», March, 1995.

[12] ref. a DCOM

[13] Harold Elliotte Rusty, «Java Network Programming» OREILLY

[14] «Sun MicroSystems Java», http://java.sun.com

[15] Decker y Hirshfield, «Programming Java, An introduction to programming using Java» PWS publishing company, Boston 1998

[16] Vogel y Keith, «JavaTM Programming with CORBA», Second Edition, Editorial Wiley.

[17] ref. TCL

[18] ref. Sistemas basados en DCOM y CORBA

[19] Comer Douglas E., «TCP/IP», 3ra edici[on, Prentice Hall, 1996

[20] Rumbaugh, Blaha, Premertani, Eddy y Lorensen, «Object Oriented Modeling & Design», Prentice Hall International, ISBN 0-13-629841-9

[21] University of Twente, «Distributed Systems», Second printing, Mullender Editors, Addison Welsey, Amsterdam, 1993

[22] Coleman y Khanna, «Groupware», International Prentice Hall, 1995.

[23] L. M. del Pino, «Realidad Virtual», Editorial Paraninfo

[24] Chorofas Dimistris N. y Steinmann Heinrich, «Realidad Virtual, Aplicaciones Practicas en los negocios y la industria», Prentice Hall

[25] Gradechi Joe, «Realidad Virtual, Construccion de proyectos», Ed. Wiley

[26] Lavroff Nicholas, «Mundos Virtuales, realidad Virtual y ciberespacio», Ed. Anaya

[27] Larijani L. Casey, «Realidad Virtual», Mc graw Hill

[28] «Realidad Virtual», http://www.cg.tuwien.ac.at/research/TR/96/TR-186-2-96-06Abstract.html

[29] «Realidad Virtual», http://www.cc.gatech.edu/gvu/people/Masters/Rob.Kooper

[30] «Realidad Virtual» http://vr.isdale.com/index.html

[31] «Realidad Virtual», http://isdale.com/jerry/VR/WhatIsVR/frames/WhatIsVR4.: & http://isdale.com/jerry/VR/techReview.html

[32] Kolski C., «Interfaces homme-machine», Editions Hermès, Paris, 1997

[33] Johansen Robert, «Groupware», Groupware92, pp 210, CA; Morgan Kaufmann

[34] Bio Control Systems, Inc. Eyecon & Brainman Products, http://www.lionhrtpub.com/ISR/ISRsubs/ISR-6.7-93/serious.html

[35] Mark Pesce, «VRML Browsing and Building Cyberspace». Prentice Hall 1997

[36] Bernie Roehl, Justin Couch, Cindy Reed-Ballreich, Tim Rohaly, Geoff Brown «Late Night, VRML 2.0 with Java» ZD PRESS

[37] Jed Hartman, Josie Wernecke «The VRML 2.0 Handbook » Silicon Graphics, Inc.

[38] «VRML», http://tecfa.unige.ch/guides/vrml/vrmlman/node26.html

[39] «Java3D», http://www.tomco.net/~raf/java3d.html

[40] «Java3D», http://www.sun.com/desktop/java3d/collateral/index.html

[41] «Java3D», http://www.sun.com/desktop/java3d/collateral/index.html

[42] «Java3D», http://www.sun.com/desktop/java3d/

[43] ref. 3D Studio Max

[44] ref. Truespace

[45] «3DGPL», http://www.cs.mcgill.ca/~savs

[46] ref. «CyberTown»

[47] «DIVE», http://www.sics.se/dive

[48] «DIVA», http://ptolemy.eecs.berkeley.edu/diva

[49] ref. «Fly3D»

[50] «Genesis 3D», http://www.genesis3d.com

[51] «Legus», http://www.legus3d.com

[52] Félix Ramos, David Garduño, Iván Romero, José Luis Márquez, « Arquitectura de una Oficina Virtual Distribuida», eMemories of the CIE98 congress, CINVESTAV México, 1998.

[53] Félix F. Ramos C., David R. Garduño B. Iván Romero. «A multiagent Based architecture for a Distributed Virtual Office», Memories of the Parallel and Distributed Processing Techniques and Applications PDPTA 99, Las Vegas, USA, Jun 1999.

[54] H. Van Dyke Parunak, «Technologies for virtual enterprises», Agility Journal, Industrial Technology Institute, Ann Arbor, MI, 1997

[55] Shinkuro Honda, Hiranori Tomioka, Takaaki Kimura, Takaharu Ohsawa, Kenichi Okada and Yutaka Matsushita, «A virtual office environment based on a shared room realizing awareness space and transmitting awareness information», Keiko University, Japan, UIST 97, Banff Alberta Canada

[56] Okada y Matsushita, «Learning from TV Programs: VideoConferencing System», ACM UIST'95, 1996

[57] Okada, Maeda, Ichikawa, Matsushita, «Multiparty video conferencing at virtual socual distance: MAJIC design», ACM CSCW'94, Chapel Hill, USA, pp 385-393, 1994.

[58] Stefik M. et all, «Beyoud the Chak Board: Computer Support for Collaboration & Problem solving in meetings», ACM, vol. 30, 1997

[59] Tang y Rua, «Montage: Providing Teleproximity for Distributed Groups», ACM CHI94, 1994.

[60] Sohlenkamp & Chwelos, «Intgrity Communications, Cooperation and Awareness: DIVA», ACM, CSCW'94, 1994

[61] Nakahashi, Yoshida Nishimura e Ishida, «Freewalk: Supporting Casual Meetings in Network», CSCW'96, 1996

[62] Greenhalg y Benfold, «Massive: A collaborative Virtual environments for teleproximity», ACM HCI No. 3, 1995.

[63] Denis Amselem, «A window in shared virtual environments», Virtual Perception Program, SRI International, Universite Joseph Fourier, Grenoble, France, On Presence: Teleoperators and Virtual Environments, pp. 130-145, 1995

[64] Roehl Bernie, «Distributed virtual reality – An overview», http://sunee.uwaterloo.ca/~broehl/distrib.html and http://ece.uwaterloo.ca/~broehl/distrib.html, June, 1995.

[65] Roehl Bernie, «Some thoughts on behavior in VR systems», http://sunee.uwaterloo.ca/~broehl/behav.html and http://ece.uwaterloo.ca/~broehl/behav.html, June, 1995

[66] ref. software de agentes

[67] ref. OMT

[68] ref. metodología Fusion

[69] Jourdon Edward, «Análisis Estructurado Moderno», Prentice Hall, 1993

[70] ref. Platinum

[71] ref. Rarional Rose

[72] Pressman Roger, «Ingeniería de Software», 3ra. Edición, McGraw Hill, 1993

[73] Boehm B. W., «Software Risk Management», IEEE Press, 1989.

[74] Gilb T., «Principles of Software Engineering Management», Addison-Wesley, 1988

[75] Boehm, «A Spiral Model for Software Development and Enhancement», Computer, vol. 21, no. 5, May 1998, pp. 61-72.

[76] Milner Robert, «Communication & Concurrency», Prentice Hall

[77] fuente de logica temporal y espacial con Kena

[78] Hofmeister, Nord y Sodi, «Software Architecture Applications», Addison Wesley

[79] ref. de origen de HTML

[80] «OpenGL», http://www.opengl.org

[81] ref. investigadores del DF que estandarizaron las funciones de una oficina virtual

[82] ref. RMI

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: "Ambiente Genérico Virtual Distribuido: GeDA-3D" de la Srita. Silvia Inés Toscano Garibay, el día 04 de octubre de 2000.

EL JURADO

Dr. José Luis Leyva Montiel
Investigador Cinvestav 3B
CINVESTAV DEL IPN
Guadalajara.

Dr. Félix Francisco Ramos Corchado
Investigador Cinvestav 2A
CINVESTAV DEL IPN
Guadalajara.

Dr. Aurelio López López
Profesor Investigador
Departamento de Maestria en
Ciencias Computacionales
INAOE, Puebla