UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

UN ESPACIO DE TRABAJO COMPARTIDO COMO APOYO A LA EDICIÓN COLABORATIVA EN INTERNET

T E S I S

para obtener el grado de

MAESTRO EN CIENCIAS

p r e s e n t a:

HENRY AFRANIO PEREZ LUNA

DIRECTOR DE TESIS: DR. MANUEL ROMERO SALCEDO

MÉXICO, D.F. Octubre 2001

Agradecimientos

A los doctores Alejandro Rodríguez Valdés y Arturo Palacio Pérez, por todo su apoyo y su constante motivación para culminar con éxito esta etapa académica y profesional.

Al Instituto de Ingeniería de la UNAM, por la beca otorgada y todos los recursos facilitados para la realización de mis estudios.

A los doctores Hanna Oktaba, Manuel Romero Salcedo, Gerardo Vega y Luis Pineda, así como al maestro Carlos Rodríguez Contreras, por sus valiosas contribuciones para la realización de esta investigación.

Al Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas de la UNAM, por brindarme la oportunidad de realizar esta especialización.

A mis amigos Sonia Mendoza, Jorge López Velarde y Martín De Jesús Jiménez, por toda su ayuda y su colaboración.

Resumen

Desde junio de 1999, se puso en marcha en el Departamento de Ciencias de la Computación, del Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS), el proyecto titulado **"Ambiente Multipropósito para la Edición Colaborativa en Intranet e Internet"**. En este proyecto se estudian diversas infraestructuras para desarrollar sistemas que faciliten la colaboración y la edición de documentos en conjunto, tomando como base un editor colaborativo en Internet llamado Alliance.

El presente trabajo de investigación se enmarca dentro del objetivo de este proyecto y presenta el desarrollo, teórico y práctico, de un espacio de trabajo compartido representado por computadoras, como un medio apropiado para facilitar el trabajo colaborativo. Dicho espacio de trabajo se ha concebido con base en las limitaciones que presenta actualmente el editor Alliance, las cuales están relacionadas con los mecanismos para la colaboración en tiempo real.

Dentro del espacio de trabajo compartido para Alliance se proporcionan herramientas para la discusión, la votación y el intercambio de mensajes, en forma síncrona. Con estas herramientas se busca apoyar a los autores que colaboran en la edición de un documento, para que lleven a cabo actividades como la negociación, la cual constituye una de las fases fundamentales en el proceso de edición colaborativa.

Esta investigación presenta inicialmente un estado del arte de los espacios de trabajo compartidos y las características del proceso de edición de documentos, utilizando Internet como el medio para la comunicación. Con ello, se presenta el contexto de los estudios que buscan determinar mejores formas de colaboración entre las personas, mediante el uso de computadoras, y las diversas propuestas de arquitecturas, herramientas y modelos, que constituyen la base para el desarrollo de sistemas colaborativos en Internet y la Web.

En el desarrollo del espacio de trabajo compartido para Alliance se aplica la metodología orientada a objetos unificada[1], utilizando el lenguaje UML (*"Unified Modeling Language"*) para el análisis de los requerimientos, el diseño de los componentes y las especificaciones para la construcción. Asimismo, este desarrollo se apoya en la definición de una arquitectura para compartir información en Internet, construida con el lenguaje de programación Java y la interface de programa de aplicación (API) Java Shared Data Toolkit (JSDT), como soporte para la comunicación de grupos.

Finalmente, con esta investigación se aportan elementos importantes que sirven de base para los desarrollos de mecanismos y herramientas, que exploren nuevas alternativas para facilitar el trabajo colaborativo y su aplicación en los proyectos del Area de Sistemas Distribuidos y Colaborativos, del Departamento de Ciencias de la Computación del IIMAS.

[1] La metodología unificada recopila las características del proceso de modelado y diseño orientado a objetos, propuesto por Ivar Jacobson, Grady Booch y James Rumbaugh, de Rational Software Corporation, http://www.rational.com

INDICE

TABLAS

Página

FIGURAS

Capítulo 1

Introducción

Los sistemas colaborativos han evolucionado notablemente en los últimos 10 años hacia la necesidad de apoyar y facilitar las actividades en grupo que realiza el hombre. Dicha necesidad ha llegado a ser un elemento fundamental en el diseño de sistemas que apoyen a las personas, en la realización de su trabajo en forma conjunta y colaborativa.

El desarrollo de este tipo de sistemas ha sido posible gracias a los avances tecnológicos en las redes de comunicaciones y a que se ha adoptado la infraestructura de la Web como la base para su construcción.

El **Trabajo Colaborativo Asistido por Computadora (CSCW**[1]**)** es el área de investigación en la cual se agrupan los desarrollos de sistemas colaborativos que soportan las actividades y el trabajo en grupo. En forma paralela, las investigaciones en el área de la **Interacción Humano Computadora (HCI**[2]**)** han aportado fundamentos para el desarrollo de estos sistemas, haciendo énfasis sobre los impactos y los beneficios que tiene el uso de computadoras en las actividades del hombre.

La edición como actividad colaborativa se ha consolidado como un tema de investigación dentro de CSCW, conocido como **Edición Colaborativa Asistida por Computadora (CSCWriting**[3]**)**. Alliance [Rom98] es un buen ejemplo de un editor colaborativo que facilita a varios autores, ubicados en sitios geográficos distantes, producir documentos estructurados en forma conjunta, usando parte de la tecnología de la Web como medio de comunicación.

En los sistemas colaborativos se han desarrollado diversas propuestas para facilitar la interacción entre los usuarios, en diversos instantes de tiempo y en espacios de trabajo comunes o diferentes. Con esta interacción se busca que un usuario adquiera un mayor entendimiento de las actividades que realizan sus demás colegas, haciendo más eficiente la realización de tareas en conjunto. Los espacios de trabajo compartidos han surgido como los medios más apropiados para apoyar esta interacción y facilitar la comunicación requerida para el trabajo en grupo.

En la presente investigación se experimenta con el análisis, el diseño y la construcción de un espacio de trabajo compartido, que sirva de apoyo al proceso de edición colaborativa en Internet. Este espacio facilita el trabajo colaborativo mediante mecanismos de conciencia de grupo e incorpora herramientas para la negociación entre los autores, con el fin de mejorar la forma como producen sus documentos.

---

[1] Del término *"Computer Supported Cooperative Work"*.
[2] Del término *"Human-Computer Interaction"*
[3] Del término *"Computer Supported Cooperative Writing"*

1.1. Definición del problema

La experiencia obtenida en los diversos desarrollos de los sistemas colaborativos en CSCW, ha permitido identificar la información que debe proveerse a los participantes del trabajo en grupo, para facilitarles el entendimiento de sus interacciones y las actividades que realizan en conjunto. Esta información, que se generaliza con el concepto de **conciencia de grupo**, no ha sido considerada como elemento determinante en el diseño de los sistemas y las herramientas de trabajo en grupo actuales.

En la presente investigación se analiza el tipo de información que debe proporcionarse a las personas para apoyar y fomentar el trabajo colaborativo, así como las diversas formas para representar dicha información, con el fin de mejorar los mecanismos de colaboración que provee el editor Alliance.

El objetivo de esta investigación es desarrollar un espacio de trabajo compartido representado por computadoras, que facilite el intercambio de la información que se requiere en actividades como la negociación, proporcionando un mayor grado de interacción y de conciencia de grupo, entre los autores que colaboran en la edición de un documento.

Para el análisis y el diseño de este espacio de trabajo, se utiliza el lenguaje UML de la metodología orientada a objetos unificada, el cual define una serie de productos (diagramas de clases, diagramas de secuencias, etc.) para facilitar la especificación de los requerimientos y el comportamiento de un sistema colaborativo.

El desarrollo del espacio de trabajo representado por computadoras, se apoya en la definición de una arquitectura para compartir información a través de Internet, construida con el lenguaje de programación Java. Este lenguaje incorpora un conjunto bien definido de librerías para el manejo de datos compartidos y permite que el espacio de trabajo sea portable sobre diversas plataformas y sistemas operativos (Windows, Solaris, Linux y Mac).

1.2. Terminología

A continuación, se definen los conceptos y términos claves que son abordados a lo largo de la presente investigación:

- Edición colaborativa en Internet

Se refiere a las actividades que realizan diferentes autores, ubicados en sitios geográficos distantes, para producir documentos en forma conjunta, usando la infraestructura de Internet para su comunicación. El uso de esta infraestructura facilita el acceso a documentos remotos, a través de estándares como el protocolo para la transferencia de hipertexto (HTTP[4]).

En el proceso de edición colaborativa los autores pueden trabajar en forma simultánea o independiente sobre un documento y deben percibir sus cambios en el instante en que suceden o en instantes posteriores.

- Espacios de trabajo compartidos

Un espacio de trabajo compartido representado por computadoras, consiste en una simulación de un espacio de trabajo físico (un salón, una oficina, etc.), donde las personas pueden ver y manipular los objetos relacionados con las actividades que realizan. El término compartido

[4] Del término *"HyperText Transfer Protocol"*.

hace énfasis en que estos objetos pueden ser usados en forma simultánea por todas las personas que interactúan dentro del espacio de trabajo.

Existe una gran variedad de sistemas que simulan espacios de trabajo, los cuales varían, entre otras características, en cuanto al tamaño del espacio virtual, las dimensiones utilizadas para la representación de los objetos y del espacio mismo (2 y 3 dimensiones), y las diversas herramientas de colaboración que se proporcionan dentro de ellos.

A lo largo de la presente investigación se utiliza el concepto de **espacios de trabajo compartidos** para hacer referencia a las representaciones, por medio de computadoras, de los espacios de trabajo reales.

- Conciencia de grupo en los espacios de trabajo compartidos

El concepto de conciencia de grupo se refiere al entendimiento que un usuario puede mantener sobre el estado de su interacción con otros usuarios en un espacio de trabajo [Gut96a]. Este entendimiento abarca la información que se mantiene y que se transmite entre los usuarios sobre su ubicación, sus actividades, las intenciones relativas a las tareas que realizan y sobre el espacio mismo.

La conciencia de grupo es un elemento fundamental y determinante de la forma de interacción entre los usuarios: incrementa la eficiencia en la colaboración y facilita el uso de herramientas comunes para el trabajo en grupo [Neu94].

- Modos de interacción y comunicación

En los desarrollos en CSCW se ha clasificado la interacción de los usuarios en dos modos o tipos: **síncrona** y **asíncrona**. En el contexto de los espacios de trabajo compartidos, la interacción síncrona se refiere a que los usuarios perciben los eventos en el instante en que suceden. Por otra parte, la interacción asíncrona se refiere a que los usuarios perciben los eventos en un tiempo posterior al instante en que suceden y dicha percepción es generalmente controlada por una acción.

Dentro de este contexto, el concepto de comunicación se clasifica también en los modos síncrono y asíncrono, haciendo referencia a dos aspectos principales: la comunicación que pueden tener los usuarios en un momento dado (video, audio, texto) y la comunicación de eventos y mensajes, requerida por el sistema para apoyar la interacción. Dicha comunicación requerida por el sistema se deriva de las acciones de los usuarios y se lleva a cabo en forma transparente a la interacción entre ellos.

- Sistemas de trabajo en grupo (*"Groupware"*)

Los sistemas colaborativos son comúnmente denominados sistemas o herramientas de trabajo en grupo o Groupware. Un sistema Groupware es un conjunto de herramientas que facilitan la realización de actividades en grupo sobre tareas comunes, proporcionando mecanismos para la interacción síncrona o asíncrona de los participantes.

Estas herramientas se agrupan en diversas categorías que principalmente contemplan:

- Correo electrónico y mensajería
- Agendas y calendarios de grupos
- Reuniones virtuales
- Conferencias en tiempo real y tiempo no real

- Administración de documentos en grupo
- Flujo de trabajo

A pesar de que los sistemas Groupware facilitan las actividades en conjunto, aún no se ha contemplado en su diseño la necesidad de proveer mecanismos que fomenten la conciencia de grupo. Por ello, las investigaciones sobre los espacios de trabajo compartidos buscan extender las capacidades de los sistemas Groupware y superar sus limitaciones. En esta investigación se hace énfasis en el apoyo a la conciencia de grupo en los espacios de trabajo compartidos, sin detallar las diversas categorías de los sistemas Groupware.

- Arquitecturas de sistemas colaborativos

Los sistemas colaborativos son un caso particular de los sistemas distribuidos, donde los recursos comprenden las diversas herramientas y elementos (documentos, imágenes, video, etc.) necesarios para la colaboración. Dichos recursos generalmente residen en diferentes computadoras que están ubicadas a lo largo de una red de interconexión.

En los sistemas distribuidos existe la problemática relacionada con la transparencia y la consistencia de la información: la información distribuida debe ser percibida en forma independiente a su ubicación y los cambios ocurridos en dicha información, deben reflejarse en todos los componentes del sistema. Con relación a estos problemas se han propuesto diversas arquitecturas para los sistemas colaborativos, basadas en el modelo cliente-servidor [KrMo93] y orientadas al almacenamiento, la difusión de eventos y el uso de las herramientas de colaboración:

- **Centralizada**: en donde los recursos residen en una sola ubicación física, conocida comúnmente como servidor. En el servidor también se ubican los procesos encargados de la administración de los recursos del sistema. Los clientes solicitan al servidor los recursos que necesitan a través de llamadas a procesos remotos (RPC[5]).

- **Distribuida**: en donde los recursos son copiados (replicados) en cada ubicación que los necesita (clientes). Cada ubicación contiene procesos tanto servidor como cliente, los cuales se encargan de las peticiones de los recursos y la consistencia de sus copias.

- **Híbrida**: en donde se mezclan características de las dos arquitecturas anteriores.

Los avances en la tecnología de la Web han permitido que los sistemas colaborativos adopten su arquitectura, como una alternativa para facilitar el acceso a la información distribuida en Internet. Además, con esta tecnología se pretende estandarizar la interface de usuario de los sistemas, mediante el uso de un navegador Web. Sin embargo, esta adopción en forma total de la tecnología Web presenta diversas limitaciones, por lo que ha sido más eficiente el utilizar sólo algunos de sus componentes. En las secciones 2.3.3 y 2.3.4 se revisan estas limitaciones y las alternativas de solución.

1.3. Antecedentes

A mediados de 1999, en el Departamento de Ciencias de la Computación del Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS), de la Universidad Nacional Autónoma de México (UNAM), se puso en marcha el proyecto titulado **Ambiente Multipropósito para la Edición Colaborativa en Intranet e Internet**[6].

[5] Del término *"Remote Procedure Call"*.

[6] Proyecto de Investigación CONACYT con número de referencia: 400316-5-J32043-A.

Dentro de este proyecto se ha establecido el estudio de diversas arquitecturas para desarrollar sistemas que faciliten la colaboración, la interacción entre los autores y la edición de documentos en conjunto, tomando como base el editor colaborativo Alliance.

Actualmente, Alliance provee mecanismos que apoyan la colaboración en forma asíncrona a través de elementos gráficos de información (iconos). Estos iconos indican los eventos de cambio ocurridos y los diferentes estados en los que se puede encontrar un fragmento o un documento.

Sin embargo, Alliance no proporciona aún mecanismos para apoyar la colaboración en forma síncrona y por lo tanto los autores deben conocer previamente, suficiente información para determinar sobre cuales documentos pueden colaborar. Además, Alliance no incorpora herramientas para la negociación de los roles que los diferentes autores pudiesen tomar durante la producción de un documento, sino que cada autor, cuyo rol es de administrador, los establece en la fase inicial de la producción.

La presente investigación se enfoca en el desarrollo de herramientas síncronas que apoyen la comunicación y la negociación entre los autores. Estas herramientas se proporcionan mediante un espacio de trabajo compartido, para promover la conciencia de grupo y mejorar la producción colaborativa de documentos en Alliance.

De igual forma, en esta investigación se experimenta con una arquitectura híbrida y los elementos que se deben considerar para el desarrollo de sistemas colaborativos en Internet.

1.4. Trabajo relacionado

La comunicación de grupos, constituye la base para la construcción de un sistema colaborativo y es un elemento fundamental dentro de la problemática relacionada con la implementación de espacios de trabajo compartidos en Internet.

Actualmente, existen numerosas propuestas de modelos teóricos, arquitecturas y herramientas de desarrollo, que buscan facilitar el establecimiento de este tipo de comunicación, con el fin de apoyar e incrementar la conciencia de grupo en el trabajo colaborativo.

En herramientas como GroupKit de GroupLab[7] y TeamWave[8] SDK ("*Software Developers Kit*"), se incorporan librerías construidas en el lenguaje Tcl/Tk[9], para la manipulación de conferencias entre usuarios y la difusión de mensajes de grupos ("*multicast*"). Arquitecturas como Promondia [GaHa97] y MetaWeb [Tre97], extienden las características básicas del servidor HTTP, para incorporar la manipulación de sesiones de grupo síncronas en la Web.

Herramientas como el Java Shared Data Toolkit (JSDT[10]) incorporan las abstracciones básicas para la definición de sesiones de grupos y diversos modos para la difusión de mensajes multicast. El sistema CONCHA [Orv99] es un ejemplo de un desarrollo que utiliza ampliamente el JSDT como base para la implementación de conferencias multicast, donde se requiere la difusión de datos multimedia sobre conexiones a gran distancia.

[7] GroupLab http.// www cpsc.ucalgary.ca/Research/grouplab/home html.
[8] Teamwave http://www.teamwave.com
[9] Del término "*Tool Command Language/Tool Kit*".
[10] http://java.sun com/products/java-media/jsdt/index.html

En adición a los elementos de comunicación, de difusión multicast y de conciencia de grupo, las interfaces para la representación de los espacios de trabajo compartidos, constituyen otra importante problemática de diseño y desarrollo. Sistemas como BSCW [Ben95] y MOTU[11] son ejemplos donde los objetos de colaboración de un espacio de trabajo, son manipulados mediante estructuras de directorios comunes para todos los participantes.

Por otra parte, en sistemas como DIVE[12] se implementa un ambiente virtual colaborativo (CVE[13]) multiusuario donde las representaciones de los objetos, los usuarios y el ambiente, son hechas mediante realidad virtual. En este ambiente la información de la conciencia de grupo hace parte de la representación misma de los usuarios mediante gestos, movimientos, la interacción con las herramientas, etc. DIVE soporta diversos formatos 3D para sus representaciones y además soporta el lenguaje de modelado de realidad virtual (VRML[14]) para proveer su acceso desde la Web.

Finalmente, el desarrollo de sistemas en Internet involucra aspectos como la tolerancia a fallas y desconexiones que ocurren durante la interacción de los usuarios o durante el envío de mensajes de grupo. Estas situaciones conllevan a la utilización de protocolos que garanticen la confiabilidad y la seguridad de los mensajes transmitidos sobre los canales de comunicación.

En herramientas como Arjuna [Par95] y W3Objects [Ing95] se definen protocolos de tipo confiable que pueden ser usados para garantizar la entrega de un mensaje, principalmente en sistemas donde la disponibilidad es crítica. En el capítulo 4 se revisa este problema y las alternativas que presenta el API JSDT para la manipulación de canales confiables.

1.5. Organización

A continuación se describe la organización de la presente investigación. En este primer capítulo se presenta el planteamiento del problema, la definición de su contexto y la terminología general utilizada.

En el capítulo 2 se aborda el estado del arte de los espacios de trabajo compartidos representados por computadora y la problemática sobre la cual se establecen sus requerimientos. En la sección 2.2 se estudia la importancia de la conciencia de grupo, definiendo sus elementos y los mecanismos para fomentarla. En la sección 2.3 se analizan las arquitecturas para desarrollar sistemas colaborativos en Internet, sus ventajas y sus limitantes. Finalmente, se revisan los desarrollos más representativos en CSCW que implementan diferentes formas de espacios de trabajos compartidos.

En el capítulo 3 se analizan las características de la edición colaborativa en Internet y las arquitecturas empleadas para almacenar documentos compartidos. Asimismo, se revisan las características del editor Alliance y de otros editores colaborativos similares, que utilizan la representación de documentos compartidos como medio para facilitar la colaboración.

El capítulo 4 presenta el desarrollo del espacio de trabajo compartido para el editor colaborativo Alliance. Inicialmente, se describe el análisis de los mecanismos de colaboración y su diseño. En la sección 4.3.1 se describe la arquitectura híbrida utilizada, especificando los componentes que apoyan la negociación entre los autores. En la sección 4.4 se describen los aspectos para la construcción del espacio de trabajo. Finalmente, en la sección 4.5 se

[11] http://motu.sourceforge.net/.
[12] DIVE http://www.sics.se/dce/dive.html.
[13] Del término *"Collaborative Virtual Environment"*.
[14] Del término "*Virtual Reality Modeling Language*".

presenta la evaluación del espacio de trabajo compartido, a partir de sus características identificadas y de su desempeño en un esquema de conexión en Internet.

El capítulo 5 presenta las conclusiones sobre el desarrollo obtenido en esta investigación, así como sus contribuciones, limitaciones y las perspectivas para trabajos futuros.

1.6. Discusión

La edición colaborativa en la Web surge como una problemática dentro de CSCW y va más allá de la producción distribuida de documentos en forma colaborativa. Esta problemática abarca también las formas de interacción de los diferentes autores y el conocimiento de las actividades en grupo, para que las contribuciones de cada uno generen información más productiva y de mejor calidad.

Los espacios de trabajo compartidos constituyen una alternativa importante para promover la comunicación, la interacción y la conciencia de grupo, especialmente cuando se requiere que el trabajo colaborativo se lleve a cabo sobre conexiones a gran distancia. En dichos espacios de trabajo se busca representar las características de los espacios de trabajo reales, adoptando lugares físicos como escritorios, oficinas, salones, etc., para la construcción de las interfaces gráficas de usuario.

El entendimiento o conciencia que los usuarios tienen de las actividades de los demás es un elemento fundamental para incrementar la eficiencia del trabajo en grupo. La presente investigación se basa en este concepto para desarrollar un espacio de trabajo compartido para el editor Alliance, que proporciona herramientas síncronas para apoyar la negociación entre los autores de un documento.

En la actualidad existen diversas herramientas para el desarrollo de sistemas colaborativos en Internet, las cuales se enfocan principalmente hacia los problemas de la manipulación de objetos distribuidos, la sincronización de eventos, la difusión de mensajes y el manejo de protocolos de comunicación tolerantes a fallas.

Dentro del objetivo de esta investigación se experimenta con la herramienta JSDT y su soporte para la distribución de datos en un espacio de trabajo compartido. El JSDT no sólo tiene la ventaja de ser portable sobre una gran variedad de plataformas y sistemas operativos, sino que además provee una librería completa de funciones para el manejo de sesiones de grupo, lo cual constituye uno de los elementos esenciales en la construcción de sistemas colaborativos.

Las diversas arquitecturas de los sistemas colaborativos se diferencian en la forma como se manipulan los recursos y los objetos de colaboración. Los sistemas colaborativos desarrollados en CSCW han utilizado con mayor frecuencia una arquitectura híbrida, donde los recursos se distribuyen entre las diferentes entidades que los usan y los mecanismos para el manejo de la consistencia y la comunicación de grupos, se implementan en forma centralizada.

Capítulo 2

Espacios de trabajo compartidos en Internet

Los sistemas colaborativos tienen como objetivo facilitar las actividades que un grupo de personas lleva a cabo en la realización de una tarea en común, proporcionando herramientas automatizadas para la comunicación y la manipulación de la información compartida.

Los desarrollos de estos sistemas han tenido principalmente dos corrientes en relación con los problemas que se intentan resolver. Por una parte, se busca facilitar la comunicación de las personas mediante diversas formas de conferencia, entre pares y grupales, utilizando audio, video, mensajes de texto, etc. Por otra parte, se busca dar apoyo a las tareas específicas de grupo, como por ejemplo: el diseño de productos, el flujo de trabajo y la edición de documentos en conjunto, entre otros.

En CSCW se han desarrollado propuestas que buscan representar mediante computadoras, espacios de trabajo compartidos como los medios para la colaboración. En sistemas como TeamRooms [RoGr96] y Mushroom [Kin96][Kin97], los espacios de trabajo compartidos constituyen el medio para facilitar el uso de herramientas como visor de archivos, agenda compartida, mensajes y votación.

Existe también otro tipo de sistemas que generan espacios de trabajo implementados con dispositivos diferentes a las computadoras, como pizarrones electrónicos (*"Whiteboards"*), tablas digitales, dispositivos de punteros, equipo de Videoconferencia, etc. Trabajos como [TaKi92][DoBl93][Gre00] son ejemplos que se basan en la Teleconferencia (conferencia a distancia) de audio y video, para simular espacios de trabajo virtuales multimedia conocidos como *Media Spaces*. Estos sistemas requieren gran disponibilidad del medio de interconexión que utilizan, ya que en su mayoría son sistemas en tiempo real y utilizan protocolos adicionales a la tecnología de Internet.

Actualmente, la adopción parcial o total de la creciente tecnología de Internet y la Web, está marcando la tendencia general en el desarrollo de los sistemas colaborativos, gracias al establecimiento de estándares globales para el acceso, intercambio y representación de la información. Sin embargo, existe una clara problemática relacionada con las limitantes que surgen en esta tecnología, para satisfacer los requerimientos de los sistemas colaborativos. Es por ello que el estudio de metodologías, patrones de diseño y nuevas arquitecturas, sobre las cuales deben apoyarse dichos desarrollos, son aún tema de discusión.

El apoyo a las actividades de grupo con el uso de computadoras, tiene asociado diversos problemas relacionados principalmente con:

- la forma de coordinar el trabajo en grupo y que los participantes cuenten con la información suficiente para hacer más eficientes sus contribuciones,
- la forma de representar (interfaces para el usuario) y almacenar la información que debe ser compartida por los participantes, de manera que no se agregue mayor complejidad al trabajo colaborativo,

- la eficiencia en los medios de comunicación (redes de computadoras y protocolos) para soportar la interacción entre participantes ubicados en sitios geográficos distantes, y
- la infraestructura (hardware, sistemas operativos, lenguajes de programación, etc.) necesaria para desarrollar estos sistemas de apoyo.

El presente capítulo se concentra en el grupo de sistemas asistidos por computadora que implementan espacios de trabajo compartidos en Internet, algunos de los cuales combinan técnicas de conferencia con herramientas específicas para ello. Inicialmente se revisan las características y requerimientos de los espacios de trabajo compartidos y las arquitecturas para sistemas colaborativos que apoyan su implementación. Finalmente, se sintetizan los sistemas más representativos en CSCW, que han influenciado el desarrollo propuesto en esta investigación.

2.1. Características y requerimientos de los espacios de trabajo compartidos

Un espacio de trabajo compartido representado por computadoras, consiste en una simulación de un espacio de trabajo físico (un escritorio, un salón, una oficina, etc.), donde las personas pueden ver y manipular los objetos relacionados con las actividades que realizan. El término compartido hace énfasis en que estos objetos pueden ser usados en forma simultánea por todas las personas que interactúan dentro del espacio de trabajo.

Los desarrollos de espacios de trabajo compartidos en CSCW buscan facilitar la interacción entre un grupo de personas, mediante interfaces visuales que les provean la sensación de estar trabajando en el mismo lugar, como sucede en un espacio de trabajo real. En estas interfaces de usuario se busca proveer la misma funcionalidad de los sistemas conformados por dispositivos diferentes a las computadoras (pizarrones, tablas digitales, etc.) y explorar diversos tipos de comunicación en medios sujetos a fallas como Internet.

Los espacios de trabajo compartidos tienen asociados diversos problemas en relación con la información que se debe manipular, representar e intercambiar, para hacer más eficiente el trabajo colaborativo. En las siguientes secciones se revisan estos problemas y las características de los espacios de trabajo compartidos.

2.1.1. Formas de interacción

El principal objetivo de los espacios de trabajo compartidos es facilitar la forma como los usuarios pueden colaborar entre sí. En el mundo real, la mayor interacción entre las personas se presenta en las situaciones donde se encuentran frente a frente en la misma ubicación física ("*face-to-face*"). La colaboración mediante esta interacción es natural, espontánea y sin limitantes o restricciones en la comunicación.

Los espacios de trabajo compartidos buscan facilitar formas de interacción similares a la forma frente a frente, mediante el uso de herramientas de colaboración. Estas herramientas adoptan mecanismos de comunicación en forma síncrona o asíncrona, para que las acciones de los participantes sean percibidas en el instante en que suceden o en instantes posteriores.

En este contexto, la comunicación no sólo se refiere a la información con la cual se trabaja conjuntamente, sino también a la información involucrada en el proceso de colaboración, como por ejemplo el estado de la interacción entre los participantes.

De acuerdo con [Ell91] la interacción en el trabajo colaborativo puede clasificarse en las dimensiones de tiempo y espacio, lo cual establece una clasificación para las herramientas que apoyan dicha interacción. La tabla 2.1 presenta algunos ejemplos de esta clasificación:

	Mismo Tiempo (Síncrona)	Diferente Tiempo (Asíncrona)
Mismo Espacio (Local)	Apoyo a la toma de decisiones Reuniones electrónicas frente a frente Pizarrones electrónicos Edición colaborativa	Pizarrón de notas Edición colaborativa
Espacio diferente (Distribuida)	Videoconferencia Teléfono Enseñanza a distancia Conferencia en tiempo real Edición colaborativa Votación	Conferencia en tiempo no real e-mail Newsgroup Flujo de trabajo Agenda electrónica Edición colaborativa

Tabla 2.1. Clasificación de herramientas de colaboración síncronas y asíncronas.

En los espacios de trabajo compartidos el concepto de interacción síncrona es más general que el de interacción asíncrona: la interacción síncrona implica una comunicación con retardos de tiempo mínimos y la difusión de eventos debe ocurrir en forma transparente a las acciones de los usuarios. Sin embargo, no en todos los casos es más conveniente, ni eficiente, facilitar este tipo de interacción, especialmente cuando se utilizan medios de comunicación sujetos a fallas como lo es Internet.

Es por ello que en los espacios de trabajo compartidos se combinan los dos modos de interacción, facilitando a los usuarios la percepción de acciones y eventos de manera inmediata, o el trabajar en forma aislada y posteriormente recuperar el estado actualizado del espacio, mediante una acción voluntaria.

2.1.2. Coordinación de eventos

La interacción entre los usuarios debe apoyarse en procesos que coordinen las actividades grupales, con el fin de resolver conflictos de comunicación de mensajes y eventos, evitar la inconsistencia de los objetos compartidos y mantener la disponibilidad del contenido del espacio mismo (persistencia de los objetos). Asimismo, independientemente de la forma como se lleve a cabo la interacción dentro del espacio (síncrona o asíncrona), los eventos deben ser coordinados de forma que garanticen el control de la concurrencia.

El intercambio de la información que fluye dentro del espacio se logra mediante la adopción de técnicas de difusión de mensajes para la comunicación de grupos. Esta difusión puede darse entre pares de usuarios (*"unicast"*), entre grupos (*"multicast"*) o el total de los usuarios (*"broadcast"*) del espacio.

El proceso de coordinación de mensajes involucra el uso de protocolos de ordenamiento para determinar la secuencia en que los eventos deben ser recibidos, atendidos y posteriormente difundidos. La figura 2.1 ejemplifica en forma simple una situación de ordenamiento de eventos.

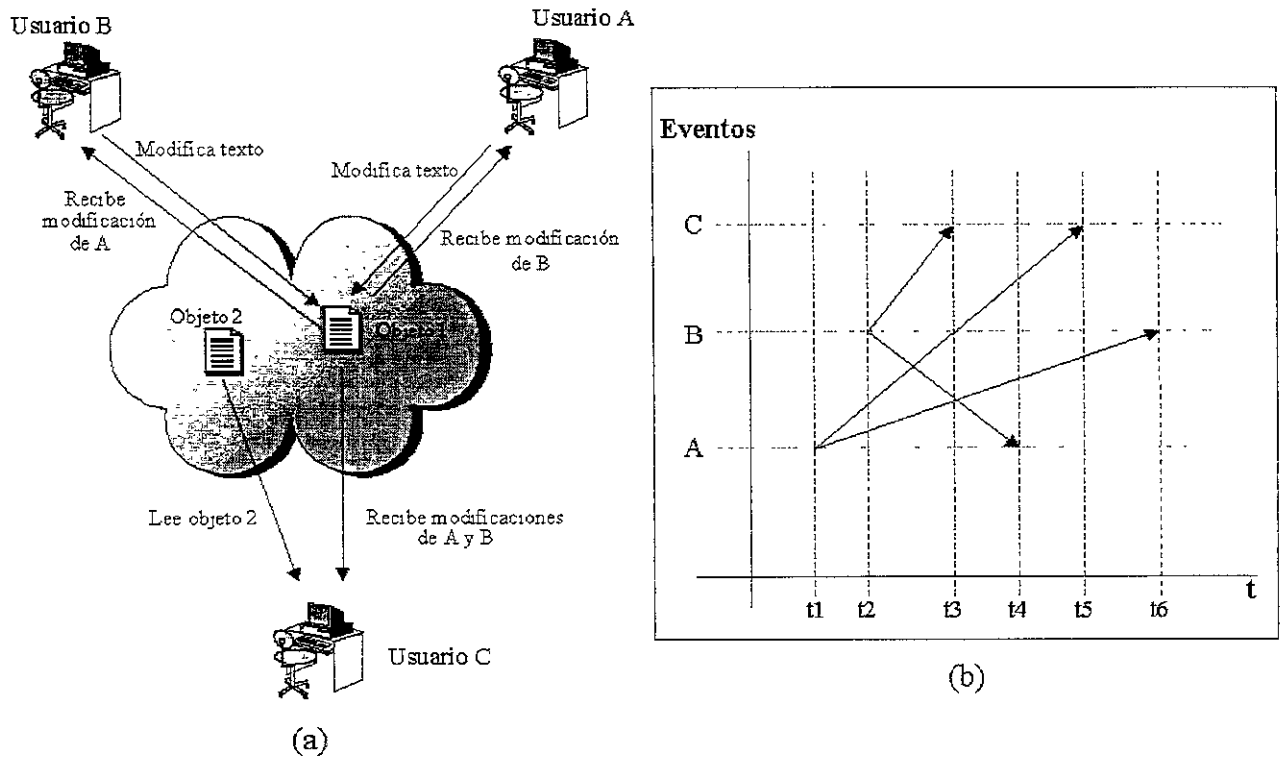


Figura 2.1. Ejemplo del problema de ordenamiento de eventos.

En la parte (a) de la figura 2.1 se ejemplifica una situación en un espacio de trabajo donde los usuarios A y B realizan modificaciones sobre el objeto 1 simultáneamente. Dicho objeto también está siendo accesado por el usuario C. En la parte (b) de la figura se muestra una de las posibles formas en que se emiten y reciben los eventos de los usuarios a través del tiempo. Nótese que A difunde su evento de modificación en el instante de tiempo t1 y B en el instante t2. Sin embargo, C recibe primero el evento difundido por B (en t3) y luego el difundido por A (en t5), asumiendo que B y C están ubicados en sitios más cercanos.

Si C sólo realiza lecturas del objeto 1, esta situación no es un problema puesto que los cambios en este objeto serán percibidos por C en algún tiempo t. Pero si las acciones de C sobre el objeto 1 dependen de los cambios de A y posteriormente de los de B, existe un problema de consistencia en el orden en que fueron emitidos y recibidos todos los eventos.

Los protocolos para el ordenamiento de mensajes y el control de la concurrencia constituyen un amplio campo de investigación, tanto en sistemas colaborativos como en sistemas distribuidos. En el contexto del presente trabajo no se realiza un análisis más detallado de estos problemas, pero sí se revisan las propiedades de ordenamiento de mensajes del API JSDT, para el desarrollo propuesto en el capítulo 4.

2.1.3. Tamaño e Interface

Las representaciones de los espacios compartidos se basan en las características existentes en los espacios de trabajo reales, por lo que comúnmente se adoptan los conceptos[1] de escritorio, salón o ambiente, para la construcción de las interfaces gráficas de usuario.

[1] Conocidos como **Metáforas** en la literatura de CSCW.

Las investigaciones en CSCW sobre el contenido de un espacio de trabajo compartido han permitido identificar elementos fundamentales que deben contemplarse en dichas interfaces:

- **Los objetos de colaboración**, que comprenden principalmente documentos de tipo multimedia para diversos propósitos.
- **Las herramientas que pueden utilizarse en forma conjunta**, como por ejemplo editores gráficos, visualizadores de archivos, envío de mensajes, etc.
- **La información sobre los usuarios**, activos o inactivos dentro del espacio, lo cual comprende información propia de cada uno y las actividades que realizan.
- **Los eventos ocurridos en el espacio**, con lo cual se incrementa el grado de interacción entre los participantes.

Las propuestas de interfaces para representar espacios de trabajo compartidos varían desde estructuras de directorios de objetos (como el explorador de archivos de Windows), hasta ambientes complejos que incorporan realidad virtual. Otra tendencia importante es utilizar la interface propia de la Web (un navegador convencional), haciendo uso de su capacidad para el acceso a la información ubicada en sitios distantes. Sin embargo, como se verá más adelante en las secciones 2.3.3 y 2.3.4, la adopción de la Web como interface, implica extender su arquitectura básica para agregarle mecanismos de interacción en forma síncrona.

Otra característica esencial en la representación de un espacio compartido es la necesidad de que su interface refleje un estado coherente del espacio mismo. Por ejemplo, cuando se manipulan objetos remotos, los usuarios deben percibirlos como si fueran objetos locales para cada uno. De igual forma, si un nuevo objeto es incorporado, dicho objeto debe reflejarse en las interfaces de todos los usuarios del espacio. Para la coherencia de la interface se ha definido el concepto de **vistas del espacio**, las cuales se clasifican en tres tipos:

- **WYSIWIS** (*"What You See Is What I See"*) [Ste87], en donde la vista del contenido del espacio es la misma para cada usuario.
- **WYSIWYG** (*"What You See Is What You Get"*) [App87], en donde la vista representa exactamente cada objeto, tal y como existe dentro del espacio.
- **WYSIWID** (*"What You See Is What I Do"*) [Gut96a], en donde cada usuario visualiza, en forma síncrona o asíncrona, las acciones realizadas por sus demás colegas.

Finalmente, existe una característica adicional referente a la representación de las herramientas de colaboración, las cuales no deben agregar complejidad a las acciones realizadas dentro del espacio. Dependiendo del tipo de herramienta, será necesario difundir a los participantes el estado de su utilización para facilitar la interacción entre ellos. Por ejemplo, el uso de una herramienta de mensajes o de votación podría requerir la atención de los participantes para generar nuevos eventos en el espacio.

2.2. La conciencia de grupo en los espacios de trabajo compartidos

Dentro del trabajo en grupo existen dos elementos fundamentales que garantizan el éxito del proceso de colaboración en la realización de una actividad en conjunto. Estos elementos se refieren al tipo de información que se comparte entre los participantes y el conocimiento de las actividades grupales e individuales que se llevan a cabo.

Cualquiera que sea el dominio de la tarea en común, las actividades grupales implican siempre un proceso de coordinación que se deriva de la interacción de cada participante. El entendimiento que puede mantenerse sobre las actividades individuales y en conjunto, es definido como **conciencia de grupo**, la cual provee el contexto para garantizar que las contribuciones individuales son relevantes para la actividad de todo el grupo [DoBe92].

Los sistemas que implementan espacios de trabajo como medios de colaboración emplean dos formas para proveer la información de la conciencia de grupo. Por un lado, se utilizan mecanismos para proveer información que se genera en forma explícita e independiente de los objetos compartidos de trabajo, como por ejemplo los perfiles (características) de los usuarios y sus ubicaciones.

Por otro lado, se utilizan mecanismos para recolectar y distribuir pasivamente la información que se genera de la interacción entre los usuarios. Además, dicha información se representa dentro del espacio de trabajo compartido como objetos de colaboración. Estos mecanismos se implementan principalmente mediante formas de comunicación síncrona y, en menor proporción, combinando formas de comunicación asíncrona.

La conciencia de grupo en la interacción frente a frente es relativamente fácil de mantener: la comunicación es directa y la percepción de lo que realizan todos los participantes es hasta cierto punto completa. Por ejemplo, en una reunión en una mesa de trabajo, todos los participantes están cara a cara con los demás y pueden discutir, intercambiar información, percibir gestos, movimientos, etc. Toda esta información fluye en forma inmediata y constituye una realimentación a cada participante en situaciones como la anticipación o la deducción de los eventos que pueden ocurrir.

Por el contrario, el mantener y proveer conciencia de grupo en un espacio de trabajo compartido es mucho más complejo. Esto se debe a que la representación por medio de computadoras, de toda la información que fluye durante una interacción en grupo, es excesivamente limitada. Estudios como [GuGr99][Gut96a] explican algunas de las principales razones que causan estas limitantes:

- Lo que las personas pueden apreciar ampliamente en un espacio de trabajo físico, se limita a vistas reducidas del espacio virtual, representadas en la pantalla de la computadora.
- Los medios de comunicación como las manos son representados débilmente por movimientos de punteros de ratón. Asimismo, la percepción auditiva pierde mucha de su calidad cuando los usuarios se encuentran en sitios distantes.
- La interacción dentro de un espacio de trabajo representado por computadora genera mucho menos información que la interacción frente a frente. En el mundo real, gran cantidad de la información que produce conciencia de grupo proviene de la interacción directa con los objetos de colaboración.
- Las técnicas de representación de usuarios dentro del espacio, como el video por ejemplo, están limitadas por la resolución de los dispositivos utilizados.

El representar la información que genere conciencia de grupo es en la actualidad un aspecto bastante discutido en CSCW. Sin embargo, se han logrado desarrollar mecanismos que facilitan el entendimiento de la ubicación de los usuarios dentro de un espacio, sus acciones y sus intenciones cuando trabajan en forma conjunta. En la sección 2.2.2 se revisan los mecanismos más comunes para representar la información de conciencia de grupo.

2.2.1. Elementos de conciencia de grupo

La conciencia de grupo en los espacios de trabajo compartidos es adquirida mediante elementos relacionados con diversas modalidades de conciencia, los cuales determinan la información que debe fluir entre los participantes:

- Quién interactúa con quién y qué están haciendo (**Conciencia Informal**).
- Qué conoce una persona de la otra en un contexto conversacional, por ejemplo: su estado emocional, el interés o la atención prestada (**Conciencia Social**).

- Los roles, las responsabilidades y la posición de una persona frente a algo (**Conciencia de Grupo Estructural**).
- La razón por la cual trabajan y comparten el conocimiento (**Conciencia Organizacional**).
- El propósito de una tarea, los requerimientos y las metas específicas del grupo (**Conciencia de Tarea**).

En estudios como [DoBe92][Gut96a][Mit95][GuGr99][Men00], se proponen otras modalidades de conciencia de grupo, las cuales complementan los elementos mencionados anteriormente y establecen los dos criterios claves para implementar la conciencia de grupo en los espacios de trabajo compartidos: la información que debe capturarse acerca de la interacción de los participantes en el espacio y como debe ser presentada dicha información.

Los principales elementos de conciencia de grupo se asocian a preguntas relevantes que cada participante debe hacerse a sí mismo durante su trabajo en grupo. Estos elementos se resumen en la siguiente tabla:

Elemento	Pregunta Relevante
Identidad	Quién está participando en la actividad?
Ubicación	Donde están los participantes?
Nivel de Actividad	Están activos los participantes en el espacio? Qué tan rápido están trabajando?
Acciones	Qué están haciendo los participantes? Cuáles son sus actividades y tareas actuales?
Intenciones	Qué van a hacer? Dónde van a estar?
Cambios	Qué cambios están haciendo? Dónde están haciendo dichos cambios?
Objetos	Qué objetos están usando?
Extensión	Qué pueden ver?
Habilidad	Qué pueden hacer?
Esfera de Influencia	Dónde tienen efecto sus intervenciones?
Expectativas	Que necesitan de mi para hacer lo siguiente?

Tabla 2.2. Elementos de conciencia de grupo.
Fuente: Traducido de [Gut96a].

Algunos de los elementos de la tabla 2.2, como la identidad y la ubicación, son relativamente fáciles de implementar mediante listas de usuarios o registros de ubicación. Los demás elementos implican mayor complejidad en su implementación, ya que involucran diversos procesos, como por ejemplo difundir a todos los participantes la ocurrencia de un evento o los cambios sucedidos en los objetos.

2.2.2. Mecanismos de conciencia de grupo

Adicionalmente a la identificación de los elementos de conciencia, es necesario entender como se lleva a cabo el proceso de adquisición de la información que fluye dentro del trabajo en grupo. Este entendimiento se refiere a como las personas obtienen la información que actualiza su estado de conocimiento.

En la interacción frente a frente existe un gran número de mecanismos de adquisición de información, que varían desde visualización de acciones, expresiones o gestos de los participantes, hasta el contacto físico con un objeto de colaboración.

Partiendo de esta base, se ha tratado de generalizar los mecanismos de conciencia de grupo con el fin de facilitar su diseño e implementación en un sistema colaborativo. Tales mecanismos se han definido ampliamente en [Gut96a] y se resumen a continuación:

- **Comunicación directa**: la información sobre la interacción con el espacio de trabajo es comunicada explícitamente por los participantes, por ejemplo mediante comunicación verbal.
- **Producciones indirectas**: la información se comunica mediante acciones, expresiones y gestos, que no son dirigidos explícitamente, pero si son hechos públicos intencionalmente.
- **Comunicación consecuente**: visualizar el trabajo y escuchar a los demás participantes proporciona gran cantidad de información sobre su interacción con el espacio de trabajo.
- **Realimentación del espacio**: la información puede ser obtenida observando los efectos de las acciones de los participantes sobre los objetos del espacio de trabajo.

Estos mecanismos, combinados con los elementos de conciencia de grupo, constituyen la base fundamental para el diseño de herramientas que permitan mantener y proporcionar la información apropiada acerca de los participantes en un espacio de trabajo compartido. A continuación se explican estas herramientas y su utilización.

2.2.3. Herramientas de conciencia de grupo en espacios de trabajo compartidos

Los sistemas colaborativos que implementan espacios de trabajo compartidos incorporan una serie de herramientas que buscan estimular el trabajo conjunto y reducir el tiempo en la realización de una tarea.

A diferencia de las herramientas de comunicación como chats[2] o conferencia de audio y video, estas herramientas buscan proporcionar información sobre los participantes, su ubicación y el estado de su interacción en el espacio de trabajo. En la literatura, los trabajos más importantes sobre la concepción de estas herramientas son quizás los realizados por [GuGr99][Gut96a][Gut96b], en donde se generalizan con el nombre de *Widgets*[3].

Los principales Widgets desarrollados para espacios de trabajo compartidos son:

- **Vistas Radar**: corresponden a vistas miniatura del espacio de trabajo total. Estas miniaturas proveen una representación espacial de la ubicación de un participante, con la cual los demás colegas pueden darse una idea de cuales son sus actividades.
- **Vistas WYSIWIS múltiples**: se utilizan para representar con mayor detalle los objetos de cada vista de cada participante. En cada vista WYSIWIS se incorporan elementos de identidad, como nombre y foto, y de actividad, como punteros de ratón remotos (telepunteros), los cuales indican con detalle lo que cada participante está haciendo en un instante dado.
- **Vista WYSIWID**: han surgido debido a las limitantes de representación en pantallas de computadora, de las vistas WYSIWIS múltiples. Las vistas WYSIWID proveen solamente una porción limitada de la vista WYSIWIS, mostrando únicamente el contexto inmediato alrededor del telepuntero de un participante. Este Widget se basa en el hecho de que en la mayoría de las acciones llevadas a cabo en las aplicaciones gráficas se involucra el puntero como dispositivo de entrada de comandos.

[2] Chat es un término del idioma inglés que equivale a **charla** o **conversación**. En el presente trabajo de investigación se utilizan éste y otros términos en el idioma inglés, en lugar de su traducción al español, ya que así han sido adoptados en la literatura de Computación e Internet.

[3] Widget es un término que se refiere a un elemento de una interface gráfica de usuario, el cual despliega información o proporciona una forma específica para que un usuario interactúe con una aplicación o con el sistema operativo.

- **Barras de desplazamiento multiusuario**: comprenden una serie de barras de diferentes colores o texturas, las cuales indican la ubicación de los participantes en una porción determinada del espacio de trabajo. Estas barras complementan las vistas radar y las vistas WYSIWIS, presentando información más detallada sobre un participante y los objetos sobre los que está trabajando.

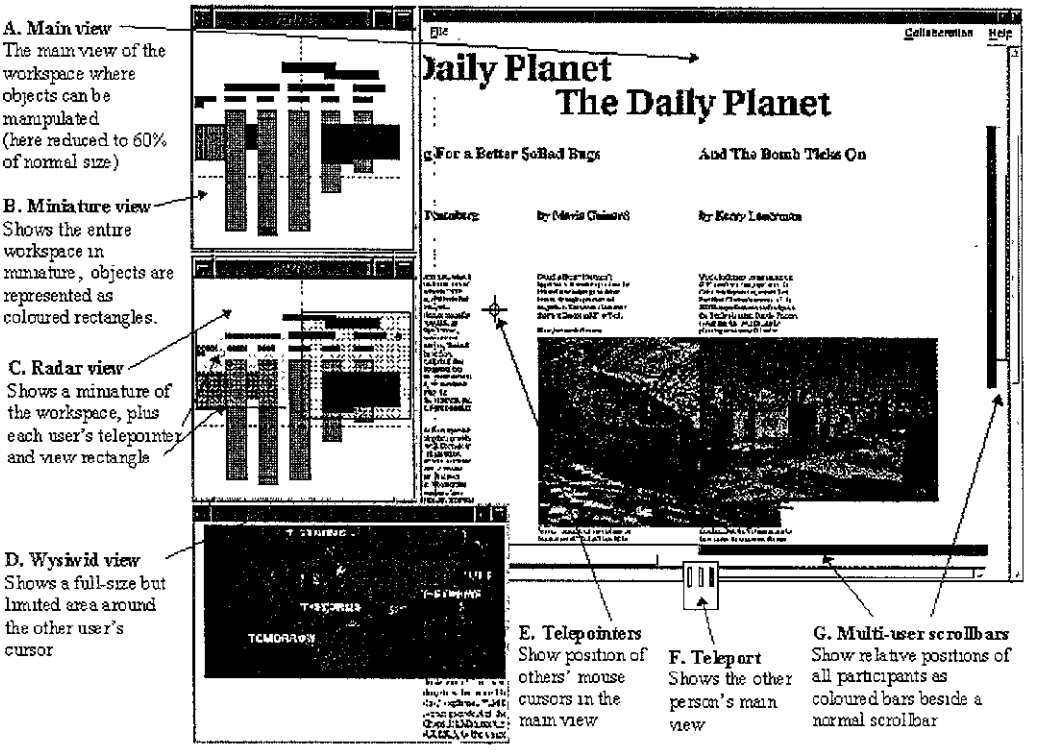


Figura 2.2. Ejemplo de interface con los principales Widgets de conciencia de grupo. Fuente: [Gut96b].

La figura 2.2 presenta un ejemplo de la combinación de los Widgets de conciencia de grupo en la interface de un documento como espacio de trabajo compartido. El recuadro A presenta la vista general del documento a manera de periódico electrónico. Los recuadros B y C presentan la vista miniatura de la ubicación de cada usuario y la vista radar de un usuario remoto en particular (nótese el recuadro sombreado). El recuadro D, presenta la vista WYSIWID del contexto de un usuario en particular. Los objetos E y G corresponden respectivamente a un telepuntero y a las barras de desplazamiento multiusuario. El objeto F (telepuerto) es un Widget específico de esta figura ejemplo, que se usa para seleccionar una de las múltiples vistas WYSIWIS de un usuario en particular.

Las herramientas descritas anteriormente han sido desarrolladas para espacios que emplean ampliamente la comunicación síncrona y buscan proveer la conciencia de grupo en los elementos de identidad, ubicación (de vista, de enfoque y de cambios) y de acción (de objetos usados, cambios realizados, nivel de actividad). Otro tipo de herramientas como listas de usuarios, registro de actividades, etc., han tratado de apoyar la conciencia de grupo en espacios que combinan la comunicación síncrona y asíncrona.

En los ambientes virtuales colaborativos (CVE) se busca que las herramientas de conciencia de grupo formen parte de la representación misma de los participantes, es decir, se busca representar información como gestos, movimientos, espacio visualizado, etc., para dar un

mayor entendimiento de la interacción de cada participante con los demás y con el ambiente de trabajo. En la sección 2.4.6 se presentan ejemplos de estas herramientas desarrolladas para un ambiente virtual colaborativo en Internet.

2.3. Arquitecturas de sistemas colaborativos

Los sistemas que representan espacios de trabajo compartidos se basan en las arquitecturas de los sistemas colaborativos y de los sistemas distribuidos para tratar diversos problemas relacionados con el diseño y la implementación.

Como se revisó en la sección anterior, la conciencia de grupo es uno de los principales elementos que deben contemplarse en el diseño de sistemas colaborativos. Por otro lado, en [DiBo99] se han establecido tres problemas adicionales que influencian el desarrollo de dichos sistemas. Estos problemas son dependientes entre sí y se definen como:

- **Comunicación**: se refiere a la necesidad de que existan enlaces interpersonales, definidos mediante técnicas y canales de comunicación, que permitan a los participantes del trabajo colaborativo intercambiar información directamente entre ellos. Dicha comunicación directa puede ser formal (planeada o programada) o informal (espontánea).
- **Coordinación**: se refiere al tratamiento y al registro de la dependencia entre las actividades de grupo y las actividades individuales. Estas actividades deben estar asistidas por mecanismos para su definición, su percepción, su seguimiento y su difusión.
- **Memoria de grupo**: se refiere al registro del proceso de la interacción del grupo, incluyendo tanto la comunicación y las tareas ejecutadas como los productos obtenidos y su historial. Los participantes deben comunicar y coordinar sus actividades teniendo acceso al conocimiento compartido almacenado en la memoria de grupo.

Los sistemas colaborativos se apoyan en arquitecturas de implementación clasificadas principalmente en tres grupos: *middleware* de sistemas distribuidos, arquitecturas de Groupware y la arquitectura de la Web.

Aunque estas arquitecturas constituyen infraestructuras completas de desarrollo, ninguna resuelve en su totalidad los problemas referidos anteriormente. Elementos como la manipulación de sesiones de grupo o mecanismos de comunicación síncrona, que incrementen la interacción entre las personas, no han sido contemplados aún como componentes básicos de estas arquitecturas y por ello continúan siendo tema de investigación. A continuación se revisan las principales características de estas arquitecturas.

2.3.1. Middleware de Sistemas Distribuidos

Middleware [Ber96], es un término que define los servicios de propósito general que se ubican en un punto intermedio entre las aplicaciones y las plataformas (sistemas operativos y hardware). Estos servicios especifican el acceso y la manipulación de objetos distribuidos en una red de interconexiones, basándose en llamadas a procedimientos remotos (RPC).

Los principales ejemplos de este tipo de arquitecturas son: CORBA[4] (*"Common Object Request Broker Architecture"*)[Wan00][Nil99], Java RMI[5] (*"Remote Method Invocation"*) y Microsoft DCOM[6] (*"Distributed Component Object Model"*)[HoKi97][Nil99].

[4] Definida por el Object Management Group (OMG), http://www.omg.org.
[5] Java RMI, http://java.sun.com/products/jdk/1.2/docs/guide/rmi/spec/rmiTOC.doc.html.
[6] Microsoft DCom, http://www.microsoft.com/NTServer/appservice/techdetails/prodarch/dcom_architecture.asp.

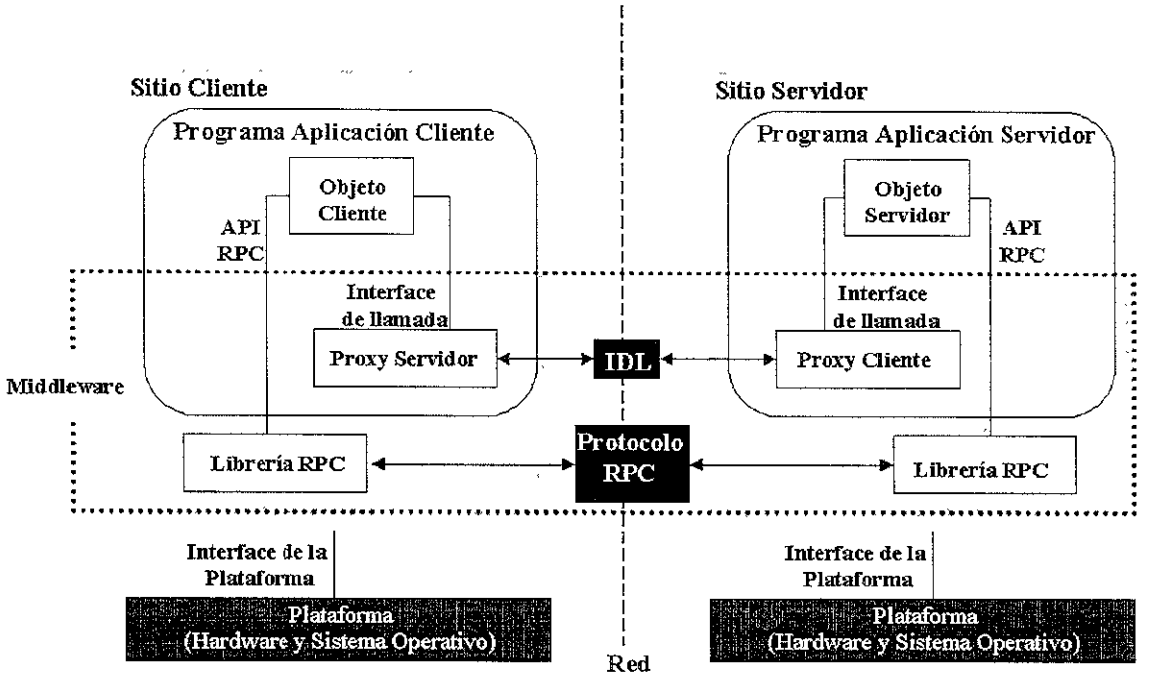


Figura 2.3. Arquitectura general middleware basada en mecanismos de RPC.

La figura 2.3 presenta la arquitectura general middleware basada en los mecanismos de RPC para el desarrollo de sistemas distribuidos. Cada sitio contiene un programa de aplicación, conformado por un objeto y sus interfaces para los mecanismos RPC. El objeto cliente es un objeto del programa del cliente que utiliza un objeto servidor. El objeto servidor es un objeto del programa del servidor, el cual es llamado desde el programa cliente (denominado componente en la arquitectura DCOM).

El *proxy* del cliente es un objeto en el programa servidor que trabaja como sustituto del objeto cliente (denominado *skeleton* en CORBA y Java RMI, y *stub* en DCOM). El proxy del servidor es un objeto del programa cliente que trabaja como sustituto del objeto servidor (denominado *stub* en CORBA y Java RMI, y *Proxy* en DCOM).

Las interfaces de los objetos proxy definen las operaciones disponibles de los objetos y como deben invocarse remotamente. Asimismo, estas interfaces especifican los procesos de empaquetamiento y desempaquetamiento (*"marshaling"* y *"unmarshaling"*), de las operaciones de los objetos y sus parámetros, para que puedan ser enviados a través de la red. Para ello, se emplea un conjunto de librerías de RPC que definen el protocolo de comunicación (específico para CORBA, Java RMI y DCOM), el cual se manipula desde cada objeto por medio del API de RPC.

Las interfaces de llamada entre los objetos del programa de aplicación y los objetos *proxy* son elaboradas en el lenguaje específico de la aplicación. Este lenguaje es definido como lenguaje de definición de la interface (IDL[7]) e involucra un compilador IDL para el desarrollo de dicha interface. En Java RMI el lenguaje IDL específico es Java. En DCOM pueden utilizarse diversos lenguajes IDL, pero debe cumplirse con estándares exclusivos de la plataforma Windows, como por ejemplo el ligado e incrustado de objetos (OLE[8]). En CORBA se soportan diversos lenguajes IDL (C, C++, Java, SmallTalk, Eiffel, entre otros) sobre diferentes plataformas.

[7] Del término *"Interface Definition Language"*.

[8] Del término *"Object Linking and Embedding"*, el cual es un estándar para manipular objetos entre aplicaciones Windows como Word, Excel, Visual Basic, etc.

En general, los principales servicios ofrecidos en las arquitecturas middleware comprenden:

- Esquema general de nombramiento de objetos para su ubicación y acceso.
- Persistencia de objetos, el cual garantiza su almacenamiento y coherencia.
- Transacción, que asegura la ejecución de las acciones que involucran varios objetos.
- Concurrencia de acceso, para garantizar el ordenamiento de invocaciones simultáneas de las operaciones sobre un mismo objeto.
- Seguridad, mediante funciones de identificación y autenticación de clientes de un objeto y la autorización sobre los métodos que pueden ser invocados.

Las arquitecturas middleware han tenido como meta la especificación de interfaces de alto nivel que reduzcan la complejidad relacionada con los procesos y protocolos de comunicación (asíncronos y síncronos) que requieren los sistemas distribuidos. Con ello, los middleware proveen soluciones ideales para sistemas distribuidos tolerantes a fallas, donde la información debe ser altamente disponible, o donde la escalabilidad y la calidad del servicio del sistema son elementos críticos.

Sin embargo, estas arquitecturas no contemplan aún problemas relacionados con la coordinación de eventos y la conciencia del trabajo en grupo, los cuales son fundamentales en el proceso de colaboración. Adicionalmente, los servicios que proveen las arquitecturas middleware no están incorporados en todos los productos o herramientas de desarrollo, por lo que es necesario integrar diversos componentes (en algunos casos de diversos fabricantes), incrementando el costo en la implementación de un sistema.

2.3.2. Arquitecturas Groupware

Los sistemas Groupware tradicionales y sus herramientas de desarrollo (*"Groupware ToolKits"*), se han enfocado en la solución de los problemas de la comunicación y la coordinación de actividades en la realización de una tarea en conjunto. De igual forma, estos sistemas abordan el problema de la memoria de grupo en los aspectos del almacenamiento y la distribución de los objetos utilizados en el proceso de colaboración.

La principal limitante de estos sistemas consiste en que aún no contemplan los elementos fundamentales de conciencia de grupo como requerimiento clave de su diseño. Por esta razón, han surgido los sistemas colaborativos que extienden las características de las arquitecturas Groupware para incrementar la interacción entre los usuarios (apoyados por computadoras) más que la interacción usuario-computadora. Dichas arquitecturas Groupware se clasifican en Groupware síncrono y Groupware asíncrono.

La arquitectura de Groupware asíncrono define mecanismos de comunicación y de interacción entre las personas en diferentes instantes de tiempo. Los objetos de información se almacenan en forma centralizada o se distribuyen secuencialmente entre los participantes, a medida que son requeridos. Bajo un esquema de interacción asíncrona no se concibe la idea de concurrencia sobre un objeto de información sino que el objeto es manipulado por cada participante en instantes de tiempo diferentes.

Asimismo, el proceso de coordinar la interacción entre los participantes resulta ser menos complejo de mantener e implementar. Por ejemplo, en situaciones como el flujo de trabajo, cada participante interactúa con los demás cuando tiene el control sobre el objeto de información o cuando concluye sus acciones y transfiere el control a otro participante. De esta forma, no es requerido ningún mecanismo de ordenamiento de eventos puesto que todos son difundidos de manera secuencial por cada participante.

Estas características asíncronas han sido extendidas en algunos sistemas Groupware para proveer cierto grado de concurrencia. El mejor ejemplo de estos sistemas es Lotus Notes[9], el cual incorpora mecanismos de bloqueo básicos para que diversos participantes manipulen un objeto en forma simultánea. Además, Lotus Notes proporciona un sistema de mensajes en línea (similar a un chat) y una agenda de grupo, con los cuales los participantes pueden comunicarse entre sí en forma semi-síncrona mediante mensajes de texto.

Bajo un esquema de comunicación en forma asíncrona resulta ser ineficiente el facilitar una interacción de grupo similar a la de tipo frente a frente. Esto se debe a que en la interacción frente a frente se requiere comunicación directa e inmediata y las acciones de los participantes no deben depender de ningún tipo de secuencia. El éxito de un proceso de colaboración involucra un alto porcentaje de actividades de discusión y negociación que requieren comunicación en tiempo real, o al menos con retardos de tiempo mínimos.

Este propósito ha motivado la exploración de arquitecturas síncronas para la comunicación de grupos, la notificación de eventos y la representación de los elementos que dan un mayor entendimiento del trabajo que realiza cada participante simultáneamente. En CSCW existe un gran número de Toolkits que proponen arquitecturas para desarrollar sistemas Groupware síncronos. GroupKit[10], NCSA Habanero[11], Egret[12], Clock[13], COAST[14], Dream Team[15] y W3Objects[16] son algunos ejemplos de estos Toolkits.

En general, en los sistemas Groupware y los Toolkits la colaboración de forma síncrona se establece sobre el elemento básico de la comunicación de grupos, definido como sesión [GrRo99][GaHa97]. En las sesiones se lleva un registro de cada participante con información suficiente para determinar su actividad. Este registro es la base para establecer el proceso de coordinación de los participantes activos dentro de la sesión. Las sesiones se implementan mediante interfaces de usuario que adoptan los conceptos de **conferencias (reuniones)**, **escritorios**, **salones** o **ambientes virtuales**.

Los sistemas basados en conferencias están diseñados para la comunicación punto a punto y multipunto, pero su desempeño decrece drásticamente cuando el número de participantes es muy alto. Además, estos sistemas incluyen herramientas básicas de comunicación (en texto, audio y video) pero no incorporan herramientas que puedan usarse en forma conjunta por varios participantes (más de dos), lo cual limita la capacidad de interacción entre ellos. Microsoft NetMeeting[17] y Netscape Conference[18] son los ejemplos más populares de este tipo de sistemas.

Los sistemas basados en escritorios y en salones proveen medios más apropiados para la coordinación del trabajo y el uso de herramientas en grupo, ya que facilitan la interacción dentro de un mismo espacio virtual. La representación de espacios mediante escritorios generalmente limita el número de participantes que pueden interactuar entre sí. La representación mediante salones proporciona diversos contextos de colaboración y la

[9] IBM Lotus Notes, http://www.lotus.com/home.nsf/welcome/notes.
[10] http://www.cpsc.ucalgary.ca/projects/grouplab/groupkit/
[11] http://havefun.ncsa.uiuc.edu/habanero/.
[12] http://csdl.ics.hawaii.edu/Research/Egret/Egret.html.
[13] http://www.qucis.queensu.ca/~graham/clock.html.
[14] http://www.darmstadt.gmd.de/publish/ocean/activities/internal/coast.html.
[15] http://carmen.fernuni-hagen.de/dreamteam/dreamteam_eng.html.
[16] http://arjuna.ncl.ac.uk/W3Objects/index.html.
[17] http://www.microsoft.com/windows/netmeeting/
[18] http://home.netscape.com/communicator/conference/v4.0/index.html.

información de conciencia de grupo se incrementa en forma equilibrada con el aumento del número de participantes en el espacio.

Las representaciones de ambientes virtuales extienden el concepto de salones y adicionan mayor realismo a la simulación del espacio y los objetos de colaboración. Con ello, se elimina una restricción que surge en los sistemas que utilizan dos dimensiones para la representación del espacio: el uso de un número limitado de ventanas para las herramientas de colaboración.

Las arquitecturas de Groupware síncrono se concentran también en la forma como se lleva acabo la ejecución (*"run-time"*) del sistema. Dicha ejecución se refiere principalmente a la distribución de los procesos, el despliegue de las interfaces gráficas, el intercambio de mensajes y el manejo de eventos. Para tratar estos aspectos se proponen las arquitecturas centralizada y distribuida.

Centralizada

En esta arquitectura el sistema reside en un servidor central para controlar todas las entradas y salidas de los participantes. Los procesos clientes del sistema residen en cada sitio cliente y son responsables del envío y recepción de mensajes y de requerimientos al servidor central. A su vez cada sitio cliente se encarga del despliegue de las salidas enviadas por el servidor. Esta arquitectura es conocida como difusión de despliegue centralizado (*"Display Broadcasting"*).

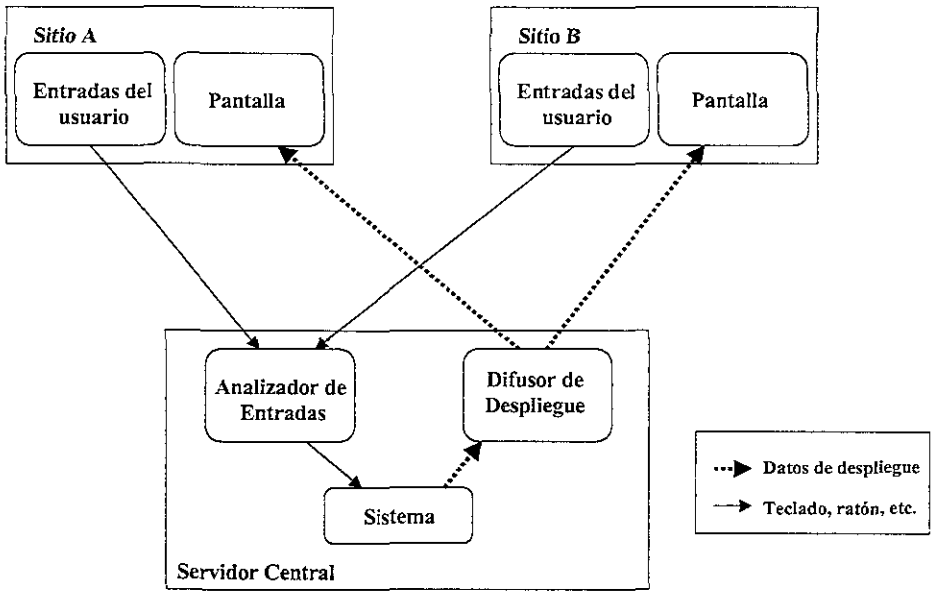


Figura 2.4. Sistema compartido síncrono con arquitectura centralizada.

La figura 2.4 ejemplifica un sistema compartido síncrono con arquitectura de despliegue y manipulación de eventos en forma centralizada. Nótese que la ejecución del sistema se efectúa solamente en el servidor central. Los clientes envían sus entradas y los eventos que generan (dispositivos como teclado, ratón, etc.) al analizador de entradas y posteriormente despliegan las salidas recibidas del servidor en sus correspondientes pantallas.

La principal ventaja de esta arquitectura comprende la facilidad para el manejo de la sincronización de los procesos: el estado del sistema es completamente consistente puesto que está ubicado en un solo lugar y los eventos son manejados por los clientes en el mismo orden en que son establecidos por el proceso servidor (serialización de eventos).

Distribuida

En la arquitectura distribuida cada cliente posee y ejecuta una copia (réplica) local del sistema compartido. Las entradas y salidas de cada cliente son difundidas a todas las copias del sistema y cada copia debe coordinar explícitamente tanto las acciones locales como las remotas. Además existe un proceso distribuido entre los clientes que se encarga de la sincronización de las copias para evitar estados inconsistentes del sistema. Esta arquitectura se denomina comúnmente difusión de eventos (*"Event Broadcasting"*).

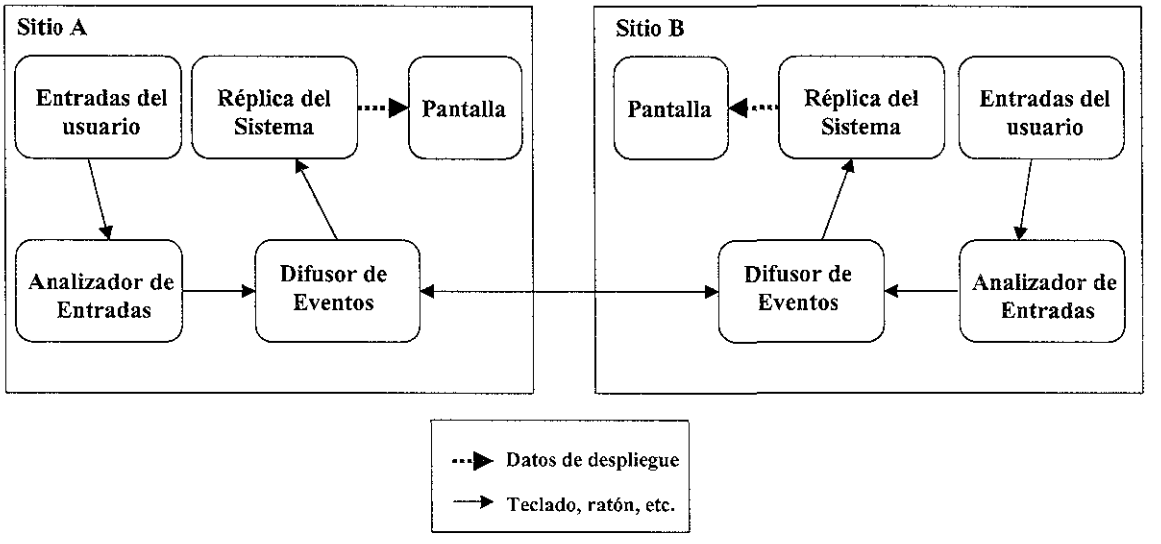


Figura 2.5. Sistema compartido síncrono con arquitectura distribuida.

La figura 2.5 ejemplifica un sistema síncrono que utiliza una arquitectura distribuida para la ejecución, la difusión de eventos y el proceso de despliegue. El proceso difusor de eventos se encarga tanto de la difusión de eventos locales como de la recepción de eventos remotos. Este proceso está distribuido en cada sitio para establecer el ordenamiento de los eventos y la sincronización entre las copias del sistema. Nótese que a diferencia de la arquitectura centralizada (figura 2.4), en esta arquitectura sólo se difunden los eventos generados por las entradas de los usuarios. Asimismo, los datos de despliegue en pantalla son generados en forma local por cada copia del sistema y no por un proceso servidor centralizado.

Debido a la facilidad de mantener un solo estado del sistema, las arquitecturas centralizadas han tenido bastante auge en el desarrollo de sistemas colaborativos; sin embargo, existen diversos limitantes:

- La arquitectura centralizada implica un esquema de procesamiento secuencial donde cada entrada de cada cliente es transmitida desde el sitio remoto hasta el servidor central. El servidor procesa la entrada y genera la respectiva salida para actualizar el despliegue de la interface en todos los clientes antes de procesar la siguiente entrada. De esta forma, el servidor establece el orden en que se procesan las entradas del sistema. Sin embargo, el establecimiento de este orden produce interrupciones y retardos en las acciones de los demás participantes, incluso en las que ocurren en forma local.

- El desempeño del sistema se convierte en un cuello de botella cuando es necesario actualizar constantemente las interfaces desplegadas, por ejemplo con el uso de vistas WYSIWIS, aumentando la carga de procesamiento en el servidor.

- El número de mensajes involucrados en el proceso de actualización de las interfaces de los clientes consume un ancho de banda considerable, lo cual afecta el desempeño en el uso de los canales de comunicación.

Una arquitectura distribuida implica procesamiento simultáneo (en los diferentes sitios), donde la manipulación de los eventos y los procesos de despliegue de interfaces pueden ocurrir al mismo tiempo en cada copia del sistema. Una implementación apropiada sugiere que solamente se transmita entre dichas copias la información respecto al estado del sistema. Así, mientras las actividades remotas pueden experimentar retardos, las actividades locales se ven ajenas a este problema y pueden ser atendidas inmediatamente. Con ello, se eliminan los cuellos de botella de procesamiento y desempeño de la arquitectura centralizada.

Sin embargo, el costo de la distribución de las copias del sistema incrementa la complejidad de su implementación y agrega la necesidad del ordenamiento de los eventos: estos ocurren en forma independiente a como deben ser procesados, generando diversos estados del sistema. Además, la difusión de eventos es un factor consumidor de tiempo, el cual debe ser consistente con el tiempo de despliegue de las interfaces gráficas de los clientes.

Las arquitecturas centralizada y distribuida han dado origen a modelos de arquitecturas híbridas donde se incorporan componentes centralizados y distribuidos. Por ejemplo, en modelos como el Servidor Centralizado de Notificaciones [Pat96], sólo los datos compartidos están centralizados en un servidor y los eventos y las vistas de los datos son distribuidos entre los clientes. Este mismo modelo está aplicado en el sistema de difusión de despliegue de X Windows de Unix, donde los procesos de despliegue gráfico están separados en cada terminal cliente-servidor del sistema.

2.3.3. Arquitectura de la World Wide Web

La *"World Wide Web"* (WWW, W3 o Web) [Ber94] se ha convertido en el medio más conveniente para accesar información distribuida en Internet. Por ello, es una excelente alternativa el adoptar la infraestructura Web para integrarle servicios de comunicación que apoyen la colaboración. Existen diversas razones que sustentan esta adopción:

- El uso de un navegador WWW integra diferentes servicios de red (HTTP, FTP, etc.) en una interface común, fácil de accesar e independiente de la plataforma.
- La Web constituye la infraestructura más utilizada por la comunidad mundial.
- El software que soporta esta infraestructura es de dominio público y extensible.

El gran auge de la Web se ha debido a la capacidad que provee para intercambiar la información distribuida globalmente mediante estándares bien definidos como HTTP, URL (*"Uniform Resource Locator"*), CGI (*"Common Gateway Interface"*), HTML (*"HyperText Markup Language"*) y XML (HTML extendido).

El HTTP es un protocolo del nivel de aplicación del modelo OSI [Bla91], para accesar información ubicada en sitios distantes, usando una arquitectura cliente-servidor [KrMo93]. En este protocolo la comunicación se establece como sigue (HTTP/1.1. versión más reciente):

1. El cliente emite un requerimiento para establecer una conexión persistente (se mantiene mientras existan requerimientos) con el servidor,
2. el servidor atiende el requerimiento,
3. el servidor responde al cliente con el resultado de la petición,
4. si existen más requerimientos sobre la misma conexión regresa al paso 2,
5. si no existen más requerimientos se libera la conexión entre el cliente y el servidor.

HTTP está basado en la familia de protocolos de comunicación TCP/IP ("*Transmission Control Protocol/Internet Protocol*"), por lo que toda la información que se intercambia en la Web es encapsulada y transmitida sobre paquetes TCP/IP.

HTTP incorpora métodos que los clientes utilizan para solicitar y obtener información de los servidores. Estos métodos están definidos como (HTTP/1.1.):

- **GET**: utilizado para recuperar objetos de información almacenados en un servidor Web o para ejecutar programas externos al servidor.
- **POST**: utilizado para ejecutar programas externos al servidor Web y el envío de parámetros para dicha ejecución. A diferencia de Get, este método involucra menos encabezados en el paso de parámetros para la ejecución de un programa (los parámetros pueden estar ocultos para el cliente).
- **PUT**: utilizado para transmitir objetos de información al servidor Web.
- **HEAD**: el cual cumple la misma función que Get pero sólo recupera la estructura o cuerpo del objeto de información.

Actualmente, el W3C[19] y el WEBDAV[20] están adicionando nuevos métodos al HTTP como Lock, Move, Copy, Delete, Authentication, etc., para hacerlo más funcional, adaptable a los requerimientos de los nuevos sistemas y proveer el control de las versiones sobre los objetos de información que se pueden manipular en la Web.

El esquema uniforme para la designación y ubicación de recursos (URL), es un estándar que permite asignar una dirección única a un objeto para su localización. Un URL está compuesto por lo siguiente:

- un protocolo de acceso (HTTP, FTP, Gopher)
- la dirección del servidor donde reside el objeto,
- el nombre del objeto.

Por ejemplo, para accesar un documento HTML ubicado en el servidor www.docs.com a través del protocolo HTTP, puede usarse el URL http://www.docs.com/documento.html.

Los lenguajes de anotaciones hipertexto HTML y XML constituyen los estándares para la representación de los documentos multimedia en la Web. Estos lenguajes utilizan el concepto de hiperliga para encadenar los diversos objetos de información, mediante el uso de URLs. En la actualidad, el HTML (4.0 la versión más reciente) soporta una serie de etiquetas especializadas para proveer acceso a otras aplicaciones externas al navegador, como por ejemplo aplicaciones de conferencia, editores de documentos en formatos diferentes a HTML (PDF, MS Office, etc.), presentaciones multimedia y demás.

El lenguaje XML extiende las funcionalidades del HTML en lo referente a la estructuración de documentos Web y el uso de nuevas etiquetas para la definición de estilos o formatos. Con este lenguaje se busca mejorar los servicios de publicación en línea y facilitar el mantenimiento de sitios Web corporativos. XML promete ser un estándar más funcional que HTML, en el cual se puede agregar mayor dinamismo al comportamiento de los documentos para incrementar la interacción de los usuarios en un sistema basado en la Web. Sin embargo, los navegadores Web actuales requieren incorporar un analizador específico de XML para soportar su uso[21].

[19] Consorcio mundial de la World Wide Web, http://www.w3.org.
[20] Grupo de Edición Distribuida y Control de Versiones en sistemas basados en la Web, http://www.webdav.org .
[21] MS Internet Explorer versión 5.5 incluye un analizador XML, pero solamente opera sobre la plataforma Windows

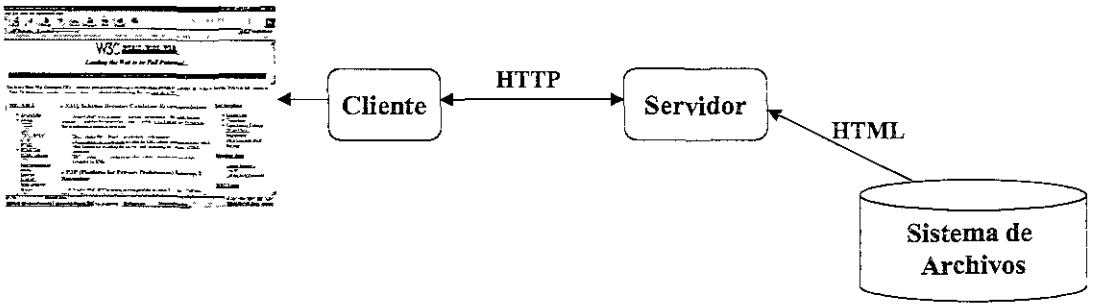


Figura 2.6. Arquitectura básica de la Web.

La figura 2.6 resume la arquitectura básica de la Web y el proceso cliente-servidor descrito anteriormente. Esta arquitectura es suficiente para el servicio de páginas estáticas (su contenido no cambia) ubicadas en servidores HTTP remotos. Sin embargo, los sistemas que requieren manipular información ubicada en servidores diferentes al HTTP (bases de datos, de procesamiento, etc.) o información que varía en su representación según los requerimientos del cliente, necesitan extender las capacidades de un servidor HTTP.

Es por ello que se ha incorporado un estándar adicional a la arquitectura básica de la Web, el cual se conoce como interface de acceso común o CGI (*"Common Gateway Interface"*) [Berl96]. Esta interface define principalmente:

- La manera en que los servidores Web se comunican con servidores externos y viceversa.
- La forma como los servidores externos generan documentos HTML para ser entregados al cliente mediante el servidor Web.

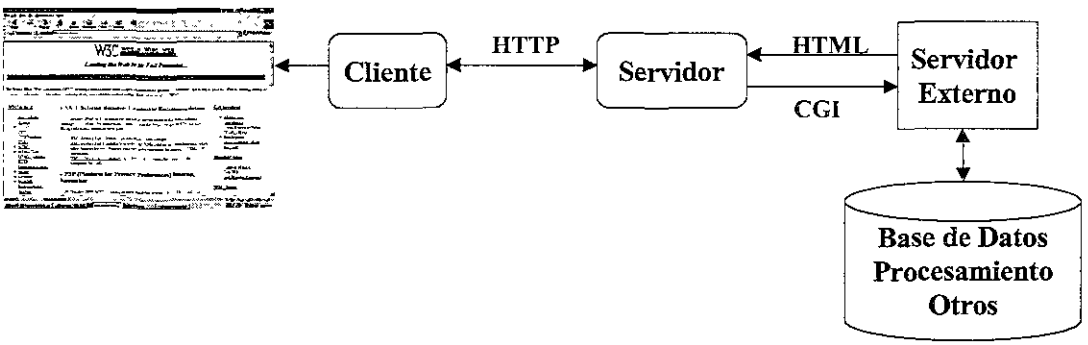


Figura 2.7. Arquitectura Web con la interface CGI.

La figura 2.7 presenta la arquitectura Web con la interface CGI para el acceso a un servidor externo específico. Actualmente, los CGI han llegado a ser muy populares en el desarrollo de interfaces Web para los sistemas de información tradicionales. Por ello, la mayoría de lenguajes de programación como Java, C/C++, Perl, Visual Basic, etc., incorporan librerías de acceso y conectividad que simplifican notablemente la programación de estas interfaces.

A pesar de la existencia de las interfaces CGI, la arquitectura Web posee una limitante muy importante para el desarrollo de sistemas colaborativos: el protocolo HTTP es un protocolo sin estado. Esto significa que bajo este protocolo ningún tipo de información es almacenada entre los requerimientos de los clientes y por lo tanto el servidor no tiene ninguna idea de cual página está siendo visitada, ni que antigüedad tienen dichas páginas en los clientes. Más aún, el cliente puede accesar otra página que está ubicada en un sitio totalmente diferente y el servidor no se entera de este cambio.

En el lado del cliente también se presenta un problema similar. El cliente sólo accesa una vista del estado del sistema en un momento dado. Si hay un cambio en el sistema, sólo se le puede transmitir al cliente hasta que éste efectúe una actualización de la página, implicando un nuevo requerimiento HTTP al servidor.

Por otro lado, el uso de interfaces CGI conlleva a que el cliente interactúe directamente con el servidor Web y no con su navegador. Es decir, el requerimiento CGI del cliente es ejecutado en el lado del servidor (nunca en el lado del cliente) y el navegador sólo sirve como medio para la representación del resultado HTML arrojado por el CGI.

Estos problemas limitan los desarrollos de sistemas colaborativos a aquellos que soportan interacción en diferentes instantes de tiempo, pero como se mencionó anteriormente, muchas de las actividades de colaboración involucran un alto porcentaje de comunicación e interacción en forma síncrona. Soluciones parciales a estas limitantes, consisten en registrar información de estado en el cliente (*"Cookies"*), para que el servidor pueda recuperarla posteriormente o trasladar parte del procesamiento del servidor al lado del cliente, mediante código incrustado en los documentos HTML, escrito en lenguajes interpretados como JavaScript o VBScript.

Sin embargo, en el caso de los Cookies, la información que puede registrarse es muy limitada y puede derivar un problema de seguridad, ya que se puede escribir información maligna en el cliente ocultando el servidor que la registra. Para el caso del código incrustado en HTML, los lenguajes mencionados son muy limitados frente a los requerimientos de sincronización y serialización de eventos que demandan los sistemas colaborativos. Por ello, la alternativa más importante para resolver las limitaciones de la arquitectura Web consiste en extender las capacidades del servidor HTTP, mediante servidores de propósito específico. A continuación se describen las propuestas más representativas de dichas extensiones.

2.3.4. Extensiones de la arquitectura Web

En desarrollos como MetaWeb [Tre97], COPSE [DiBo99], Promondia [GaHa97] y GroCo [Wal96], se extienden las funciones básicas del servidor HTTP con servidores de funciones orientadas al control y el registro de sesiones, la difusión y sincronización de eventos y la serialización de las operaciones sobre los recursos compartidos.

MetaWeb implementa un servidor que utiliza su propio protocolo de comunicación. Este servidor se encarga de la ubicación de los objetos de información, el control de la concurrencia, el control del acceso y la propagación de eventos. El protocolo de comunicación está basado en la difusión de eventos, los cuales se envían a sesiones de interés (usuarios conectados a estas sesiones), usuarios específicos o al sistema en general. Adicionalmente, se implementa una aplicación MetaWeb cliente, basada en *Applets*[22] de Java, la cual coordina la interacción con los objetos, apoyada con una interface API para la comunicación con el servidor especializado.

COPSE incorpora un proceso servidor principal (*Project Server*) que activa y desactiva procesos servidores secundarios de documentos y especiales. Los servidores de documentos se encargan de la distribución y la administración de copias de los documentos almacenados y coordinados por el proceso servidor principal. Los servidores especiales comprenden el manejo de agentes para la coordinación de mensajes, las herramientas de colaboración (calendario, flujo de trabajo, correo electrónico, etc.) y los objetos de información. COPSE utiliza una aplicación cliente especializada, desarrollada en Java, para el manejo de las sesiones de

[22] Programas compilados (clases) en lenguaje Java que se incrustan en las páginas HTML.

usuario, mediante las cuales un cliente se conecta al servidor principal o a los secundarios y utiliza las diversas herramientas de colaboración.

Promondia consiste en un programa servidor y una serie de iniciadores de sesiones, implementados como Applets de Java incrustados en documentos HTML. Cuando un Applet es inicializado, se conecta al servidor Promondia y se incorpora a una sesión específica. Si la sesión no existe se crea una nueva sesión en el servidor. Una vez se registra en la sesión, el cliente obtiene su rol y lleva a cabo la interacción con los demás participantes. El servidor Promondia es una aplicación en Java que implementa su propio servidor HTTP e incorpora un módulo para el manejo de los canales de comunicación de grupos. Además, este servidor suministra información a los clientes sobre el estado de las sesiones y sobre el control de acceso.

GroCO es un sistema basado en el modelo de reuniones electrónicas EMS[23], con el cual se establece un conjunto de conferencias para la comunicación de grupos. Las conferencias son manejadas mediante páginas Web, para el registro y la interacción de los clientes. Estas páginas comparten la información del estado de la conferencia en forma dinámica y no estática como funciona la arquitectura Web tradicional. Dichas páginas dinámicas consisten en Applets de Java incrustados en documentos HTML para la incorporación a las conferencias. Las conferencias son manejadas por un proceso coordinador de eventos, separado del servidor HTTP, que utiliza un puerto TCP/IP predefinido.

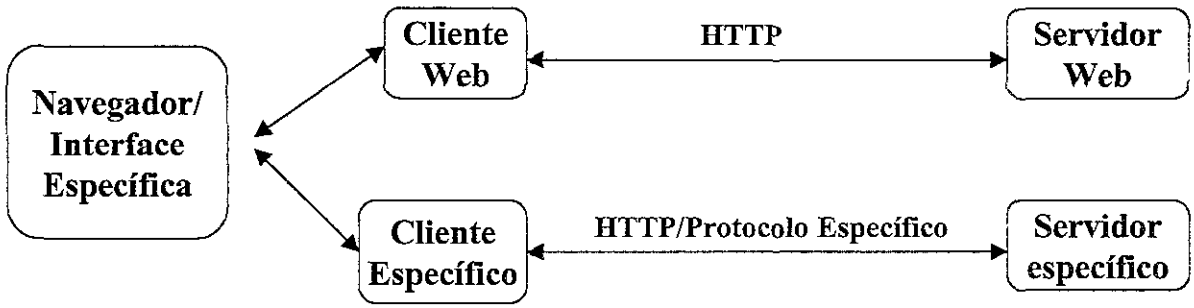


Figura 2.8. Esquema general de las extensiones del servidor HTTP.

La figura 2.8 presenta el esquema general de la extensión del servidor HTTP básico con un servidor específico, el cual atiende los requerimientos mediante HTTP o mediante su propio protocolo. La interface de usuario puede ser un navegador convencional que soporte componentes de cliente específico para la comunicación con el servidor (generalmente Applets). Algunas extensiones del servidor HTTP implementan su propia interface en lugar del navegador convencional.

El uso de Applets en documentos HTML agrega mayor capacidad de interacción con el sistema desde el lado del cliente. Sin embargo, los Applets tienen una limitante muy importante respecto a las políticas de seguridad para el desarrollo de sistemas en Internet. Java utiliza un administrador de seguridad que determina lo que un Applet puede hacer y lo que no.

Dicha política establece que un Applet no puede almacenar información en el cliente para evitar el registro de información maligna. Además, un Applet no puede realizar conexiones a un servidor diferente desde donde fue cargado, para evitar que se ignoren restricciones impuestas por *Firewalls*[24] en conexiones a sitios seguros. En los ejemplos anteriores, a

[23] Del término "*Electronic Meeting Systems*".
[24] Hardware o software que funciona como filtro para las conexiones seguras sobre Internet.

excepción de Promondia, los Applets y los servidores especializados deben residir en la misma ubicación del servidor HTTP, es decir deben estar centralizados, lo cual conlleva a los problemas como la tolerancia a fallas.

En trabajos como W3Objects [Lit97] se explora el uso de proxys (sustitutos) transaccionales como extensiones del servidor HTTP. Los clientes se conectan a procesos sustitutos del proceso servidor, los cuales no son centralizados sino que se distribuyen como servicios a través de diferentes puertos TCP/IP. La función principal de estos proxys es mantener la información del estado del sistema al lado del cliente y transmitir los eventos de los clientes a los demás servidores proxy. Esta solución requiere modificaciones al navegador Web convencional para soportar la comunicación con los procesos sustitutos.

2.4. Desarrollos de espacios de trabajo compartidos

En CSCW y HCI existe un gran número de desarrollos de sistemas colaborativos que implementan formas de espacios de trabajo compartidos para facilitar y apoyar el trabajo en grupo. Como ya se ha mencionado, estos espacios se basan en conceptos como directorios, escritorios, salones, pizarrones, oficinas y ambientes virtuales.

En el capítulo 3 se revisan algunos sistemas que utilizan la representación de un documento compartido como el espacio para la colaboración. En la siguiente sección, se describen los desarrollos que generalizan las características de los espacios de trabajo y que han servido como influencia para el desarrollo que se presenta en este trabajo de investigación.

2.4.1. Soporte Básico al Trabajo Colaborativo (BSCW)

BSCW [Ben95][HoBe97] es un sistema que implementa espacios de trabajo compartidos basados en el concepto de directorios de objetos y la publicación de documentos. Estos conceptos están implementados en desarrollos comerciales como Iplanet Web Server[25], Documentum 4i[26], MS-Internet Information Server[27], IBM-Lotus Domino[28] y Adobe Document Server[29].

Los espacios (directorios) son accesibles a los usuarios mediante un esquema simple de cuenta de acceso y contraseña. Los objetos que pueden depositarse en el espacio corresponden a documentos, imágenes, hilos de discusión asíncrona, ligas URL (sitios Web y FTP), información sobre los usuarios y anotaciones.

Dentro de cada espacio de trabajo compartido existen iconos gráficos asociados a los objetos para proporcionar información sobre su estado y sus características. En cada espacio existen diversas ligas relacionadas con las operaciones que pueden realizarse sobre los objetos según su tipo, como son entre otras: visualizar objetos multimedia, editar documentos HTML y TXT, modificar las propiedades de los objetos, crear nuevos directorios, asignar derechos sobre objetos y publicar documentos.

[25] http://www.iplanet.com/products/infrastructure/web_servers/index.html.
[26] http://www.documentum.com/products/content/index.html.
[27] http://www.microsoft.com/siteserver/default.htm.
[28] http://www.lotus.com/home.nsf/welcome/domino
[29] http://www.adobe.com/products/docserver/main.html

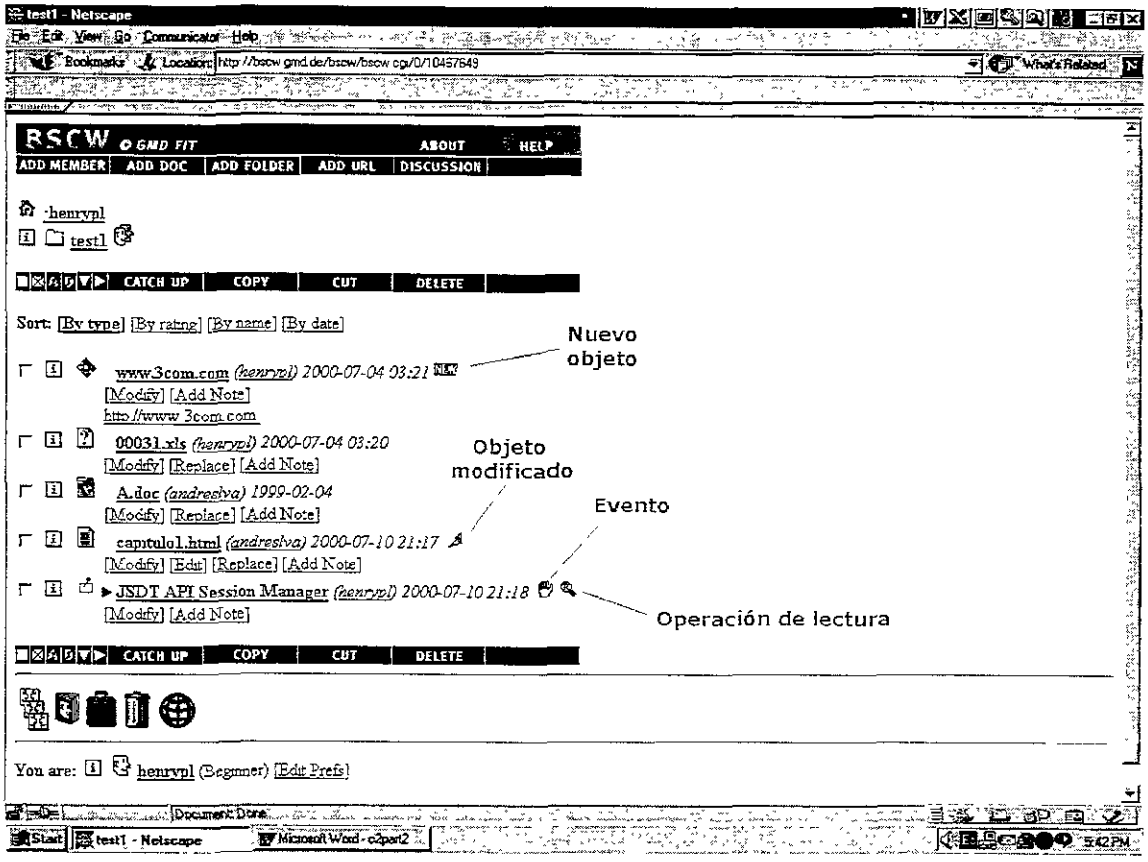


Figura 2.9. Espacio de trabajo compartido en BSCW.

La figura 2.9 presenta un ejemplo de un espacio de trabajo en BSCW, el cual se visualiza como un directorio, utilizando un navegador Web convencional. El directorio contiene una liga URL, documentos (HTML, MS-DOC y MS-XLS) y una liga de discusión sobre un tema específico. El directorio contiene también iconos que indican un objeto nuevo, un documento HTML modificado (icono lápiz), un evento ocurrido dentro de la liga de discusión (icono mano de advertencia) y una operación de lectura efectuada sobre la misma (icono de lupa).

Respecto a los objetos tipo documento, BSCW soporta el manejo de los formatos más populares como HTML, MSOffice, PDF, PostScript y TXT. Sin embargo, las operaciones de edición asíncrona concurrente, publicación y modificación de contenido, son exclusivamente sobre documentos HTML y TXT. Para el caso del formato XML, en la versión más reciente del BSCW (3.4.3, octubre del 2000), se ha implementado un navegador específico, basado en Applets de Java, el cual incorpora una vista WYSIWYG para la visualización y la edición de estos documentos.

La conciencia de grupo en BSCW se provee mediante iconos representativos que indican diversas situaciones ocurridas en el espacio como: bloqueos sobre documentos editados por los usuarios, cambios ocurridos en un objeto, nuevos objetos agregados al espacio, anotaciones, comentarios sobre objetos, etc. BSCW provee el manejo de versiones de los documentos HTML, las cuales pueden ser recuperadas, editadas, publicadas y sincronizadas para mantener la consistencia de la versión más reciente del documento.

BSCW adopta totalmente la arquitectura centralizada de la Web, basándose en la extensión MetaWeb, con lo cual permite la colaboración entre usuarios ubicados en sitios distantes, que

utilicen diversas plataformas y sistemas operativos. El servidor de Web BSCW extiende el servidor de Web básico mediante CGIs escritos en el lenguaje Python[30]. Con estos CGIs se provee la notificación de eventos en forma asíncrona y se soporta el proceso de publicación de documentos extendiendo las funciones del método *HTTP-PUT*.

A pesar de ser un sistema colaborativo en la Web, BSCW no incorpora aún mecanismos síncronos que apoyen la interacción en actividades como la negociación directa entre los usuarios. Por ello, las nuevas versiones de este sistema exploran el uso de componentes de Java que agreguen dinamismo a las páginas HTML y XML para resolver estas limitaciones.

2.4.2. Workplace

Workplace[31] es la versión comercial del prototipo Teamrooms [RoGr96], el cual es un sistema que implementa espacios de trabajo compartidos basados en el concepto de salones. Este concepto se refiere a la implementación de espacios compartidos que simulan salones físicos, donde los usuarios ubicados en sitios distantes pueden depositar objetos e interactuar con ellos en forma conjunta. Cada usuario puede accesar diversos espacios para colaborar con sus colegas sobre tareas o actividades comunes.

Los objetos que se pueden depositar en el espacio comprenden documentos, imágenes, URLs, accesos a otros salones, anotaciones y la información personal de los usuarios. Dentro del espacio existen dos herramientas compartidas que forman parte de su interface: chat de texto y pizarrón electrónico. Los usuarios utilizan estas herramientas en forma concurrente para aportar sus contribuciones y establecer comunicación en forma síncrona.

En Workplace existe también un conjunto de herramientas colaborativas de propósito específico, las cuales son incorporadas al espacio por cualquier usuario en el momento en que las necesite. Estas herramientas comprenden: visualizador de archivos, manejo de estructuras para una base de datos simple, calendario, agenda, lista de tareas, lista de actividades para reuniones, lista de direcciones, tablero de mensajes, votación de conformidad o desacuerdo sobre preguntas o comentarios y discusión de ideas.

Desde su prototipo inicial Teamrooms, Workplace incorpora herramientas para promover la conciencia de grupo en forma síncrona y asíncrona. Estas herramientas también forman parte de la interface del sistema y suministran información general y particular de las actividades que realizan los usuarios. Estas herramientas comprenden:

- Lista de usuarios conectados: es una lista que incluye información personal de cada usuario y su fotografía.
- Telepunteros: comprende un puntero de ratón de diferente color para cada usuario que se encuentra dentro del mismo espacio en el mismo tiempo.
- Vista radar del espacio: comprende un conjunto de recuadros que representan la ubicación de cada usuario dentro del espacio. Esta vista es muy útil cuando la interface de los espacios no se visualiza completamente, debido a las capacidades de resolución y el tamaño de las pantallas de las computadoras que se usan.

Cuando están trabajando en el mismo espacio al mismo tiempo, los usuarios no sólo perciben información particular de sus demás colegas, sino también perciben en tiempo real las acciones que ellos están realizando. Cuando se encuentran en espacios diferentes, los

[30] Lenguaje interpretado similar al lenguaje Perl. http://www.python.org/
[31] http://www.teamwave.com/advantages/index.html

usuarios pueden enterarse sobre sus ubicaciones, si se encuentran en modo desconectado y si se han sincronizado con el espacio.

Estas características implican la persistencia de los objetos dentro del espacio, para lo cual Workplace utiliza mecanismos de almacenamiento de datos y registro de eventos en cada cliente, controlados por un proceso servidor centralizado. Workplace ha sido diseñado con una arquitectura híbrida en la cual se distribuye en los clientes la ejecución y el almacenamiento de objetos, garantizando la persistencia de la información aún sin la presencia de ningún usuario dentro del espacio.

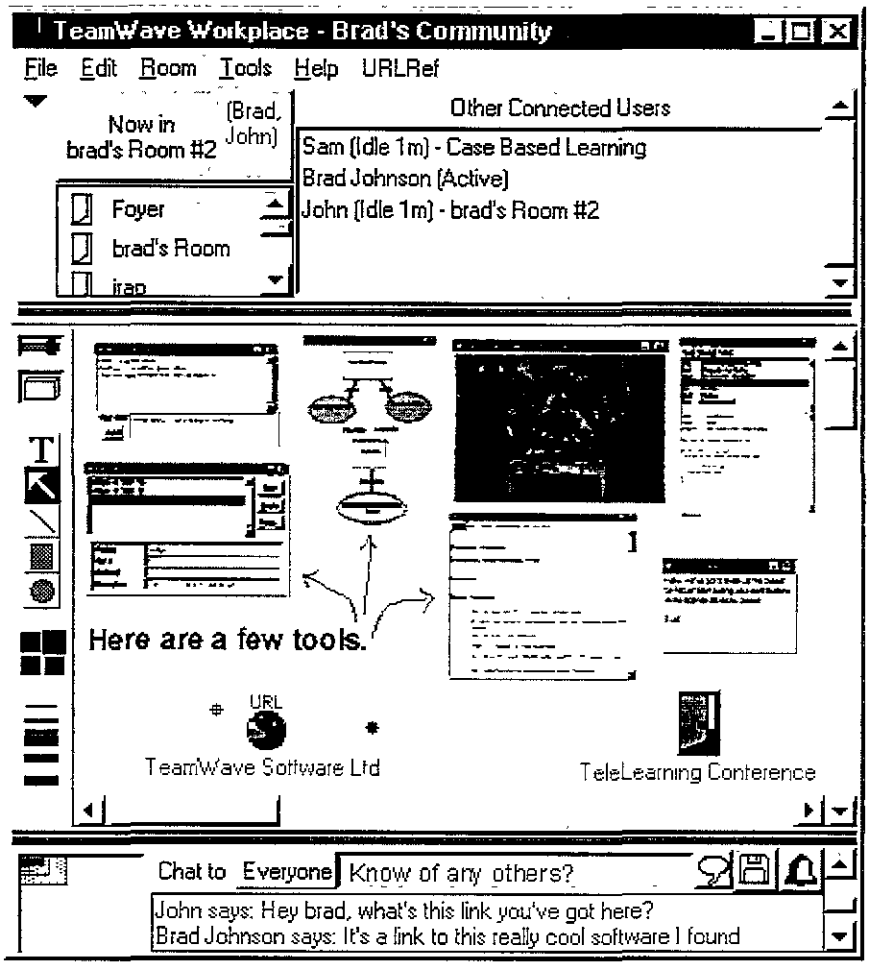


Figura 2.10. Interface de Workplace con algunas herramientas de colaboración. Fuente: www.teamwave.com.

La figura 2.10 presenta varias de las herramientas de colaboración y de conciencia de grupo proporcionadas en Workplace. El panel superior contiene la lista de los salones disponibles (parte izquierda) y la ubicación de cada usuario conectado al sistema (parte derecha). El panel intermedio es un pizarrón compartido que se visualiza con una vista tipo WYSIWIS. En el pizarrón se pueden realizar dibujos utilizando la paleta de la parte izquierda y se depositan los diversos objetos que se desean compartir.

La parte izquierda del panel inferior contiene una miniatura de la vista radar de cada usuario dentro del salón actual, con un recuadro por cada uno. Igualmente, en este panel se provee el

chat básico con opciones para envío de mensajes privados y para almacenamiento en disco del contenido de la charla. Nótese en el panel intermedio, los telepunteros de los usuarios activos en el salón Room#2 representados por los círculos a cada lado del objeto URL. Las herramientas de colaboración como agenda, votación y demás son incorporadas al espacio usando el menú principal ubicado en la parte superior de la interface.

El sistema Workplace ha sido diseñado para soportar grupos pequeños de usuarios (hasta 20) que utilizan redes de comunicación altamente confiables. Este sistema está desarrollado en lenguaje Tcl/Tk y utiliza las extensiones definidas en GroupKit para la creación de conferencias de usuarios y la distribución de eventos síncronos y asíncronos entre ellos.

2.4.3. DeskTOP

DeskTOP [Por99] es un sistema que provee un espacio de trabajo compartido para facilitar la interacción social, con la capacidad de conocer y establecer comunicación con nuevos usuarios en contextos diferentes. El objetivo en DeskTOP es proveer diversos canales de comunicación usando diversas aplicaciones. Este objetivo surge con la idea de que la interacción entre los usuarios no es predecible en la mayoría de los casos y por ello los sistemas deben ser flexibles para soportar la transición entre una interacción informal y una interacción formal.

DeskTOP se basa en los conceptos de salón, salas de colaboración y escritorios. El primer concepto hace referencia a la visión global del espacio, la cual es utilizada para la familiarización con los nuevos usuarios incorporados. El segundo concepto se refiere a cada espacio utilizado para el trabajo de un grupo de usuarios. El tercer concepto se refiere al espacio de trabajo donde los usuarios manipulan los objetos de colaboración:

- Objetos de información: documentos, libros, papeles.
- Objetos de comunicación: teléfono, fax.
- Objetos de soporte: calculadora, calendario, notas.

Las herramientas proporcionadas en DeskTOP proveen los servicios de comunicación, colaboración, coordinación y conciencia de grupo. Estas herramientas facilitan los tipos de interacción síncrona o asíncrona dependiendo de cada herramienta.

Servicio	Herramientas Síncronas	Herramientas Asíncronas
Comunicación	**Chat, pizarrón, teléfono, audio/video**	**Mail, post-it**
Colaboración	**Presentaciones, votación, editor**	**Editor, votación, calculadora, calendario, notas, reloj**
Coordinación	**Control**	**Control**
Conciencia de grupo	**Usuarios, ambiente, salón de colaboración**	**Usuarios, ambiente, Salón de colaboración**
Otros	**Asistente, escritorio**	**Tarjetas**

Tabla 2.3. Clasificación de las herramientas de colaboración en DeskTOP. Fuente: Traducido de [Por99].

La tabla 2.3 presenta la clasificación de las herramientas proporcionadas en DeskTOP. Estas herramientas son activadas por cualquier usuario dentro del espacio y son presentadas en ventanas independientes dentro de la interface de cada uno.

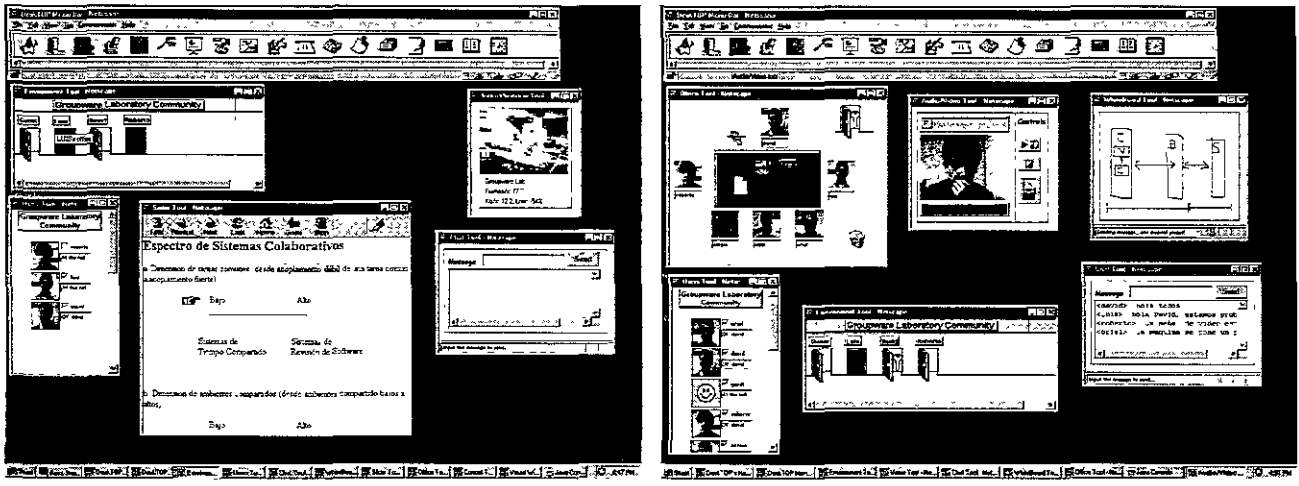


Figura 2.11. Interface del sistema DeskTOP. (a) Interface del Salón de colaboración. (b) Interface del escritorio, herramienta de ambiente y otras herramientas de colaboración. Fuente: [Por99].

La figura 2.11 ejemplifica la interface en DeskTOP. En la parte (a) se muestra una vista global del espacio, con los usuarios que se encuentran activos. En la parte (b) se muestra un ejemplo de usuarios colaborando en un escritorio y usando herramientas de video y de mensajes de texto.

DeskTOP está desarrollado con una arquitectura híbrida basada en el modelo middleware de tres capas: cliente-*broker*-servidor. La capa del cliente corresponde a la interface del sistema y de las herramientas de colaboración. La capa del servidor mantiene el contexto de los salones de colaboración y del salón global. La capa del broker actúa como intermediario entre las capas anteriores, abstrayendo los aspectos de comunicación sobre Internet: coordinación de la comunicación, envío de requerimientos, recepción de resultados y control de errores.

La interface del sistema DeskTOP está elaborada en lenguaje HTML combinado con Applets de Java, lo que lo hace un sistema altamente portable. Sin embargo, a pesar de proveer herramientas que dan apoyo a la interacción síncrona como por ejemplo chat, video y audio, DeskTOP no incorpora aún facilidades para la edición de documentos en forma concurrente y está orientado a ser principalmente un espacio virtual para la comunicación.

2.4.4. Mushrooms

Mushrooms[32] [Kin96][Kin97] es un sistema que soporta la administración y la creación dinámica de salones llamados *Mrooms*, los cuales son espacios de trabajo que contienen objetos compartidos. Estos objetos hacen referencia a las aplicaciones y los objetos de información como documentos, imágenes, audio, video y presentaciones multimedia.

En Mushroom se incorporan herramientas de colaboración que proporcionan el manejo de los objetos información. Estas herramientas abarcan la reproducción de audio y video, la comunicación de texto y la conciencia de grupo. Los usuarios que interactúan en un espacio son representados mediante listas de fotografías y la información particular de cada uno,

[32] http://www.dcs.qmw.ac.uk/research/distrib/Mushroom/.

como e-mail, URL y teléfono. En el sistema se lleva un registro de las actividades que realiza cada usuario, el cual puede ser visualizado por cualquier usuario que ingrese a un salón Mroom.

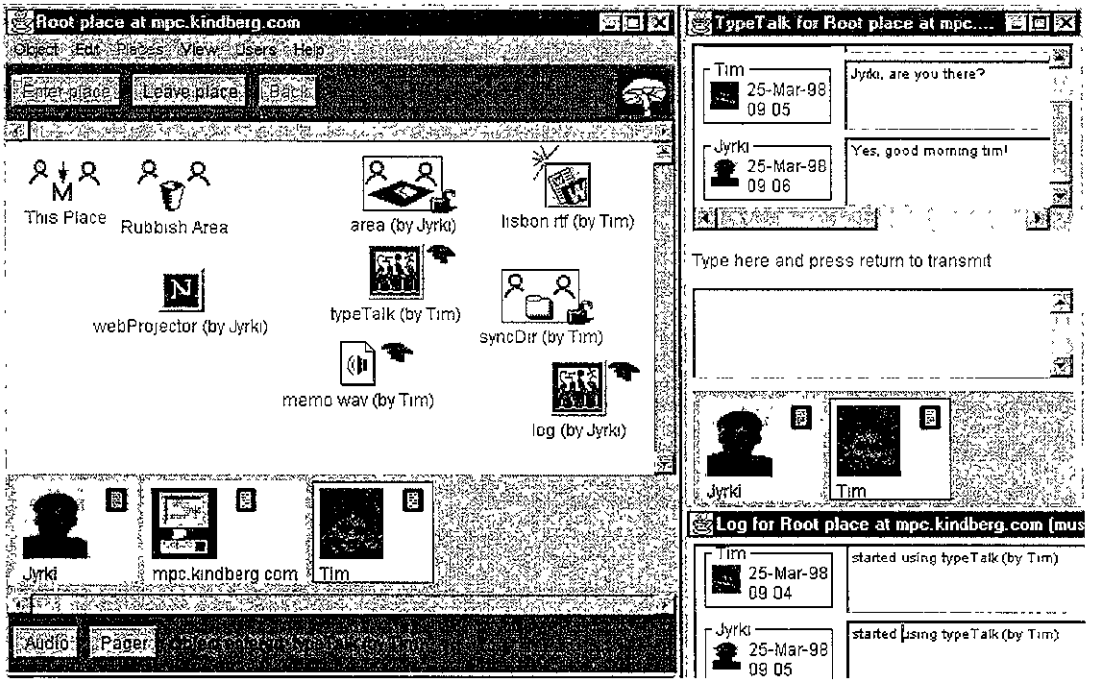


Figura 2.12. Interface de Mushroom con objetos de información y herramientas de colaboración. Fuente: www.dcs.qmul.ac.uk/research/distrib/Mushroom/indexI.html.

La figura 2.12 presenta la interface del sistema Mushroom con tres ventanas activas. La ventana de la parte izquierda presenta el menú principal del sistema y los objetos compartidos en el espacio. La ventana superior derecha presenta una herramienta de chat donde los usuarios se representan con su foto, la fecha y la hora de su conexión. La ventana inferior derecha presenta un registro de las actividades de cada usuario que está activo en el espacio.

Mushroom provee interacción entre los usuarios de manera síncrona y asíncrona. En el primer caso, los usuarios trabajan en el mismo Mroom al mismo tiempo. En el segundo caso, los usuarios trabajan en el Mroom en instantes de tiempo diferentes, pero se enteran de los cambios ocurridos en los objetos de colaboración mediante una acción de sincronización.

Independientemente del tipo de interacción que se realiza, los objetos de información son persistentes en cada Mroom, es decir, dichos objetos permanecen en el espacio aún sin la presencia de los usuarios. Los cambios sobre los objetos y los eventos en el espacio ocurren casi de manera inmediata. Mushroom provee un mecanismo de tolerancia a fallas mediante el cual un objeto almacenado es disponible a pesar de retardos o desconexiones en la red de comunicación.

Mushroom está basado en una arquitectura distribuida de ejecución y almacenamiento de objetos, mediante copias de cada uno entre las diferentes ubicaciones de los clientes. Estas copias almacenan tanto los espacios Mroom como los objetos de información contenidos en ellos. En cada cliente también se almacenan diferentes versiones, de sólo lectura, de los objetos de información y se proveen operaciones como **leer versión actual** y **obtener copia actual** (denominadas réplicas vivas) para la consistencia de los mismos.

Con el manejo de copias y de versiones de objetos, los usuarios pueden recuperar los estados anteriores de los objetos o pueden sincronizar su vista del espacio con la vista actual del grupo. Estas funciones proveen alta disponibilidad de la información de colaboración ya que en cualquier instante un usuario puede obtener la información más actualizada en el espacio, independientemente de su forma de interacción.

A pesar de que se requieren mecanismos complejos de control de concurrencia, de consistencia y de seguridad de objetos, Mushroom resuelve parcialmente el problema de la distribución de copias en sistemas a gran escala. Esta solución se basa en un protocolo de ordenamiento total de los mensajes difundidos para el manejo de las copias, el cual es un esquema optimista frente a fallas en los medios de comunicación.

2.4.5. Interfaces Dinámicas para Actividades Cooperativas (DIVA)

DIVA [SoCh94] es un sistema que implementa un ambiente de oficina virtual que reúne un conjunto de herramientas para la conciencia de grupo, la realización de tareas en conjunto y la comunicación, en los modos de interacción síncrono y asíncrono. Estas herramientas incluyen conferencia de audio y video, herramienta de edición de texto, chat, pizarrón y anotaciones.

El sistema DIVA ha sido modelado sobre los elementos físicos requeridos para la colaboración en un ambiente de oficina real: personas, salones, escritorios y documentos. Estos elementos están representados mediante objetos gráficos en dos dimensiones.

Las personas son representadas mediante pequeñas fotografías con el nombre respectivo de cada usuario. Esta representación puede ser movida y ubicada dentro de diferentes salones e indica las actividades que están realizando los usuarios y con quiénes se tiene comunicación de audio y video en un momento dado.

Los documentos representan los objetos de trabajo dentro de la oficina virtual. Estos objetos comprenden texto, imágenes, multimedia y presentaciones. Cada objeto tiene propiedades referentes a sus cambios y su estado de actividad. Además, los documentos pueden estar distribuidos en diferentes salones y para el control de la concurrencia se utiliza un mecanismo de múltiples lecturas y una sola escritura.

Los escritorios constituyen el lugar de trabajo dentro de los salones y se utilizan para almacenar el contexto de los objetos de colaboración. Los salones son los contenedores de las personas, los escritorios y los documentos.

La conciencia de grupo en DIVA se provee de varias maneras. La comunicación síncrona entre los usuarios es apoyada por el sistema de conferencia de audio y video, el cual se activa cuando un usuario arrastra el icono representativo de dicha herramienta sobre un usuario dentro de un salón. De igual forma, el usuario puede activar una herramienta de mensajes de texto para establecer comunicación asíncrona con otros usuarios.

Cada actividad que realiza cada usuario es notificada a los demás, tanto en el mismo salón como en los demás salones. Para ello, se utiliza un mecanismo distribuido de difusión de mensajes basado en el establecimiento de canales de comunicación entre cada usuario.

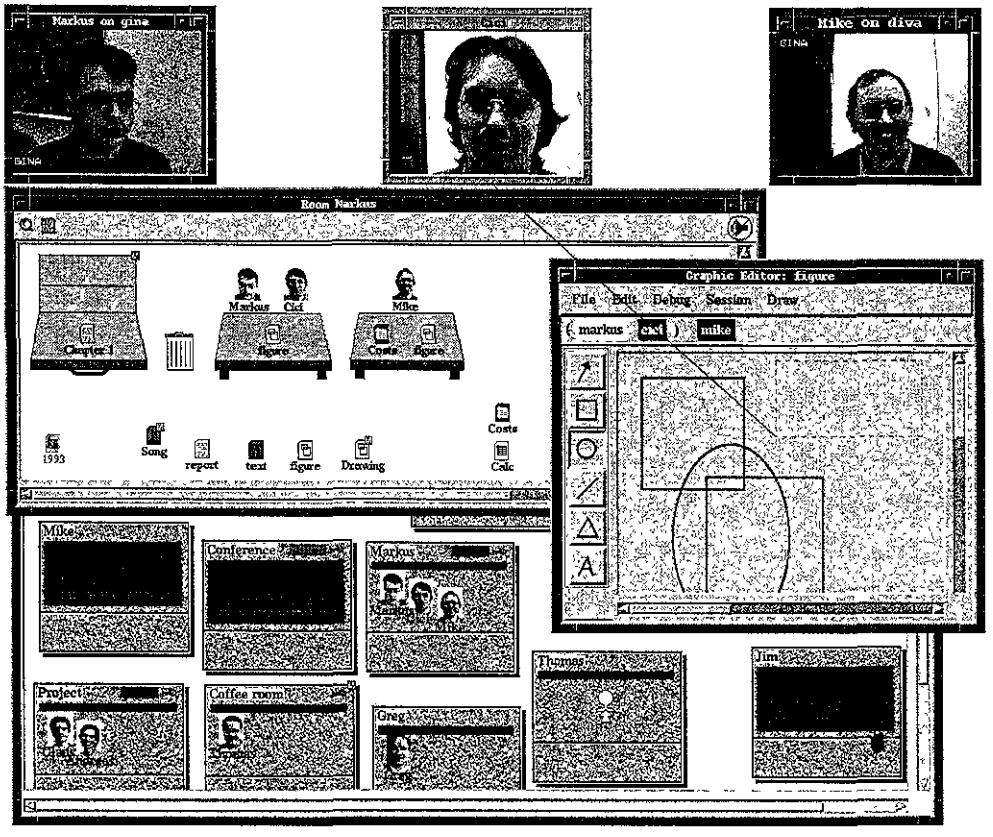


Figura 2.13. Interface del sistema DIVA con escritorios y herramientas de comunicación de video. Fuente: fit.gmd.de/projects/diva/.

En la figura 2.13 se muestra un ejemplo de la interface DIVA con diferentes usuarios que interactúan en diferentes salones y con tres usuarios en el mismo salón (dos en un mismo escritorio) que se comunican a través de la herramienta de video. Los usuarios que están en el mismo escritorio trabajan sobre el mismo diagrama y el usuario Cici (imagen central superior) es quien se encuentra realizando modificaciones.

DIVA utiliza una arquitectura distribuida y está implementado en el lenguaje Common Lisp y CLOS (*"Common Lisp Object System"*), encapsulando las acciones de los objetos y sus propiedades. Cada objeto es copiado entre los usuarios para establecer su comunicación y con ello cada acción ejecutada localmente por un usuario es ejecutada remotamente por los demás. DIVA utiliza un control de concurrencia optimista, detectando y resolviendo los conflictos de las acciones de los usuarios, mediante el uso de estampas de tiempo virtuales y un protocolo de ordenamiento de acciones conflictivas.

2.4.6. Ambiente Virtual Interactivo Distribuido (DIVE)

DIVE [FrAv98] es un sistema que implementa espacios de trabajo persistentes basados en el concepto de ambientes virtuales de tres dimensiones. Este sistema explora las representaciones de objetos y espacios mediante realidad virtual. Con ello, se superan las limitantes de las representaciones en dos dimensiones, respecto a la dificultad de manipular información de una gran cantidad de usuarios, mediante múltiples ventanas y registros de actividades dentro del espacio.

El sistema DIVE es considerado un ambiente virtual colaborativo más que un simple espacio de trabajo compartido y está diseñado para soportar aplicaciones multiusuario donde los participantes están conectados a través de Internet. Este sistema ha sido integrado a la Web mediante el uso del lenguaje para modelado de realidad virtual VRML. Sin embargo, el soporte para la Web está limitado por las fallas o interrupciones que pueden suceder en las conexiones a grandes distancias. Las representaciones mediante realidad virtual demandan tiempos de respuesta mínimos para la difusión y percepción de los eventos que suceden en los ambientes multiusuario.

Los objetos dentro del ambiente constituyen personas, lugares (cuartos, salones y pasillos), elementos físicos (sillas, mesas, paredes, tableros, etc.) y herramientas síncronas y asíncronas de colaboración. Cada persona es representada mediante un cuerpo humano simulado (Avatar) con características que lo hacen distinguible entre los demás, por ejemplo el color de la ropa y la forma del cuerpo, entre otras.

DIVE provee herramientas de colaboración como: comunicación de audio, comunicación de texto, salón virtual, pizarrón electrónico, calendario compartido, documentos, libro de notas, carpetas, anotaciones, proyector y correo electrónico. Además, un usuario puede incorporar al ambiente algún documento multimedia, el cual es interpretado automáticamente por la respectiva aplicación en que esté elaborado.

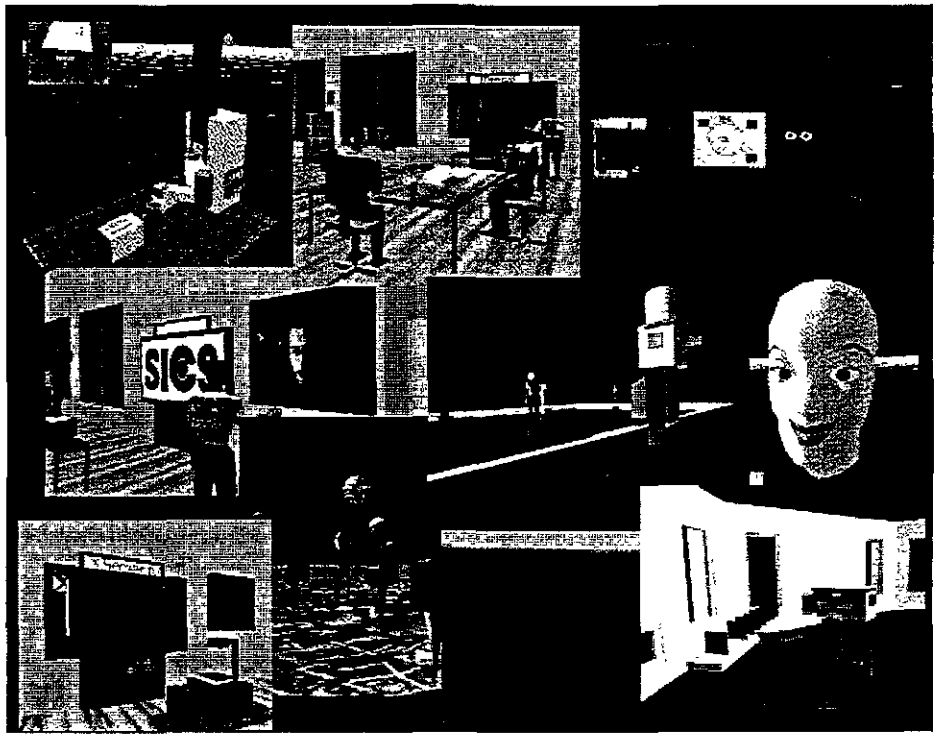


Figura 2.14. Imágenes de diversos componentes del ambiente DIVE. Fuente: www.sics.se/dce/dive.html.

La figura 2.14 presenta algunas de las imágenes que se visualizan en DIVE, como un rostro humano (cuadro a la derecha), tres Avatars en un salón (cuadro central superior), un robot en un pizarrón (cuadro superior a la derecha), un Avatar frente a una imagen de video (cuadro del centro a la izquierda) y diversas vistas de un ambiente específico.

Las representaciones mediante realidad virtual aportan una gran cantidad de información sobre lo que sucede en el ambiente y el estado de la interacción entre los usuarios, por lo que DIVE apoya ampliamente el entendimiento y la conciencia de grupo. Sin embargo, a pesar de que este tipo de representación es más ajustada al mundo real, existen situaciones en que la percepción de los objetos se limita por los mismos obstáculos existentes en el ambiente. Por ello, DIVE provee mecanismos de identificación de usuarios similares a los empleados en los espacios de trabajo compartidos como listas de usuarios y ventanas de información particular.

DIVE ha sido desarrollado con una combinación de varias herramientas y lenguajes de programación. Las interfaces del sistema y las acciones de los objetos dentro del ambiente están desarrolladas en Tcl/Tk. Algunas representaciones usan AC3D[33] y Java 3D. El soporte para la Web está desarrollado con VRML y además se utilizan Applets y JavaScript para algunas especificaciones dinámicas de las páginas HTML.

EL sistema DIVE se basa en una arquitectura totalmente distribuida, donde tanto los objetos como los eventos son copiados e interpretados por cada componente del sistema ubicado en cada sitio cliente. Adicionalmente, DIVE provee funciones de coordinación basadas en Tcl/Tk, propias de cada objeto con lo cual se resuelve la consistencia de las copias y el control de la concurrencia. Cada objeto es copiado junto con su comportamiento y cuando sucede un evento sobre él se utilizan protocolos de ordenamiento de mensajes coordinados por el mismo objeto.

2.4.7. Comparación de desarrollos

Las siguientes tablas sintetizan las principales características que presentan los desarrollos de espacios de trabajo compartidos revisados anteriormente.

Sistema	Concepto	Objetos de Información	Interface	Conciencia de Grupo
BSCW	Directorios	Documentos, URL, notas, discusión, carpetas, papelera, portafolio, imágenes	- Navegador convencional - Navegador BSCW (Java)	Iconos de notificacion asincrona
Workplace	Conferencias, salones	Documentos, imagenes, notas, ligas a salones, URL, discusion	Propietaria elaborada en Tcl/Tk	Lista de usuarios, telepunteros, vistas radar, difusion sincrona de las herramientas de colaboración
DeskTOP	Salones, escritorios	Documentos, imágenes, notas, telefono, fax, calculadora	Navegador convencional con maquina virtual de Java	Lista de usuarios, video punto a punto, audio
Mushrooms	Salones	Documentos, imágenes, presentaciones, audio, video, carpetas, ligas a salones	Propietaria elaborada con Java Applets	Lista de usuarios, audio, registro de eventos
DIVA	Salones, escritorios	Documentos, imágenes, presentaciones, video, audio	Propietaria elaborada en CLOS	Lista de usuarios, audio, video, ventanas de ubicacion
DIVE	Ambiente virtual 3D	Documentos, imagenes, presentaciones, video, audio, sillas, mesas, proyectores, pizarrones, escritorios	- Propietaria elaborada con Java, Tcl/Tk, AC3D - Navegador con soporte VRML	Lista de usuarios, audio, video, simulación virtual, movimiento, expresiones corporales

Tabla 2.4. Características generales de los desarrollos de espacios de trabajo compartidos.

Debido a que BSCW se basa en el concepto de directorios de objetos, la interacción y la conciencia de grupo que se provee a los usuarios dentro del espacio es muy limitada. Los

[33] Modelador de gráficas de 3 dimensiones, http://www.comp.lancs.ac.uk/computing/users/andy/ac3d.html.

sistemas basados en conferencias, escritorios, salones o ambientes, superan estas limitantes puesto que el espacio de trabajo es más ajustado a los espacios físicos reales y los objetos pueden ser manipulados en forma más directa.

A pesar de que los desarrollos de espacios de trabajo manipulan objetos tipo documento, en la actualidad ninguno provee facilidades de edición en forma concurrente. Esto se debe a que en estos desarrollos se busca principalmente el establecimiento de la comunicación y la representación de un espacio virtual común como medio para la colaboración.

Al igual que BSCW, DeskTOP y DIVE son sistemas altamente portables ya que su interface está basada en los navegadores convencionales, que incluyan soporte para lenguajes específicos como Java y VRML. Workplace y DIVA, son sistemas que dependen de la plataforma y requieren componentes ejecutables instalados en cada cliente para su funcionamiento.

Mushrooms es un sistema que a pesar de no tener como interface un navegador, puede ser altamente portable ya que está desarrollado para la maquina virtual de Java, la cual es independiente de la plataforma y los sistemas operativos. Dicha máquina virtual debe ser instalada en cada sitio cliente como ocurre con Workplace y DIVA.

Sistema	Herramientas		Arquitectura de Colaboración
	Síncronas	Asíncronas	
BSCW	———	Edición documentos HTML, TXT, hilo de discusión, mail	Centralizada, con extensiones del servidor HTTP mediante CGIs
Workplace	Pizarrón, vistas radar, chat telepunteros, notas, votación, visualizar archivos, base de datos, agenda, calendario, actividades	mail	Híbrida, con ejecucion y almacenamiento distribuido, coordinación y difusión de eventos con un proceso centralizado
DeskTOP	Chat, pizarrón, audio, video, telefono, editor, votación, proyector	Mail, notas, calculadora, calendario, reloj, block de notas	Híbrida, con ejecución y almacenamiento distribuido, coordinación centralizada y difusión de eventos tanto en forma centralizada como distribuida
Mushrooms	Audio, video, proyector, sincronizacion del espacio, chat	mail, presentaciones	Ejecución, almacenamiento, coordinacion y difusión de eventos mediante procesos distribuidos de alta disponibilidad
DIVA	Video, audio, pizarrón, chat	Edición de texto, mail, anotaciones	Ejecución, almacenamiento, coordinación y difusión de eventos en forma distribuida, mediante encapsulamiento de acciones en los objetos de colaboración
DIVE	Video, audio, proyectores, pizarrones	mail	Ejecucion, almacenamiento, coordinación y difusión de eventos en forma distribuida, mediante procesos de control replicados en cada cliente

Tabla 2.5. Herramientas y arquitecturas de colaboración de los desarrollos de espacios de trabajo compartidos.

Las arquitecturas de colaboración híbridas y distribuidas, empleadas en estos desarrollos, facilitan la incorporación de herramientas síncronas de colaboración, ya que la distribución del procesamiento en los clientes aumenta el desempeño que se requiere en un sistema colaborativo.

De la tabla 2.5 se concluye que el sistema BSCW es el más afectado por el hecho de adoptar la arquitectura centralizada de la Web, aún con sus extensiones, ya que sólo provee herramientas asíncronas para la colaboración y la difusión de eventos se restringe a las

acciones de los usuarios en diferentes instantes de tiempo. Mushrooms, DIVA y DIVE exploran una arquitectura distribuida total, en la que el proceso de coordinación de eventos también es distribuido en los clientes. Esto agrega mayor complejidad en el desarrollo, pero hace que el sistema sea más robusto y tolerante a los problemas comunes de particiones en la red de comunicaciones y las fallas que ocurren en un servidor centralizado.

2.5. Discusión

En el presente capítulo se estudió el contexto social que enmarca las investigaciones y los desarrollos de espacios de trabajo compartidos. Dicho contexto incide completamente en la identificación de los requerimientos de los sistemas actuales y establece los parámetros que deben contemplarse en el diseño y la implementación de la nueva generación de sistemas colaborativos en Internet.

Siguiendo la firme convicción de que los sistemas colaborativos deben adaptarse a la forma como se trabaja en grupo en el mundo real, los espacios de trabajo buscan representar los elementos con los que las personas interactúan diariamente, como documentos, escritorios, salones, personas y demás. Este proceso de interacción es un proceso bien definido en situaciones donde las personas se encuentran frente a frente, pero involucra un gran número de problemas cuando se lleva a un modelo implementado en computadoras.

Los principales problemas identificados en el desarrollo de un espacio de trabajo compartido comprenden: los procesos de comunicación entre los participantes y el espacio, la coordinación de los eventos que suceden simultáneamente en las diferentes ubicaciones, las acciones que se llevan a cabo sobre los objetos de información y el entendimiento de las actividades que el grupo de personas realiza en forma individual y en conjunto (conciencia de grupo).

El desarrollo de un sistema colaborativo debe contemplar diversos aspectos que son tratados desde diferentes contextos. En el presente capítulo se revisaron principalmente:

- El diseño de la interface de usuario, la cual debe ser clara y de tamaño suficiente para la representación del espacio y su contenido. Para ello, las vistas de tipo WYSIWIS aportan un mayor grado de entendimiento, puesto que cada usuario visualiza el mismo estado del espacio, sin necesidad de sincronizarse ni depender de mecanismos de notificaciones.

- Los elementos de conciencia de grupo que apoyen el conocimiento de las actividades grupales. Las listas de usuarios y el registro de sus actividades, son los elementos más comunes, ya que son relativamente fáciles de implementar y aportan información suficiente sobre la interacción llevada a cabo.

- Las herramientas de comunicación como chat, conferencia de audio y video, etc., que deben suministrarse dentro del espacio para apoyar la interacción en forma síncrona.

- Los mecanismos de coordinación de eventos síncronos y asíncronos que deben establecerse entre los participantes, para que el trabajo colaborativo obtenga mejores resultados.

En forma paralela a la identificación de los problemas de los sistemas colaborativos, es necesario revisar las herramientas, lenguajes de programación, protocolos de comunicación, etc., que se requieren para el desarrollo de dichos sistemas.

En esta investigación se concluye que a pesar de que existen diversas propuestas de arquitecturas para desarrollar sistemas colaborativos, ninguna puede tomarse como la receta definitiva. Sin embargo, las arquitecturas híbridas que combinan el manejo distribuido de los procesos y los eventos del sistema, y el control centralizado de los objetos de información, resultan ser las más empleadas ya que generan un mayor desempeño. No obstante, el proceso centralizado para el control de los objetos, debe en cierta forma ser distribuido en los clientes, para que pueda ser tolerante a fallas en medios como Internet.

Por otro lado, las propuestas de Toolkits para sistemas Groupware síncronos han abierto el panorama de las herramientas de comunicación que pueden ser incorporadas en un espacio de trabajo. Con ello se busca que dichos Toolkits faciliten no sólo el desarrollo, sino la integración de dichas herramientas, dentro de un sistema abierto que adopte cualquier arquitectura, especialmente la Web. Adicionalmente, como se discute en [Ise98] y [Beg97] estos Toolkits no se deben orientar solamente al desarrollo de nuevas herramientas, sino que deben permitir que una aplicación o herramienta monousuario existente se pueda adaptar fácilmente para ser colaborativa, sin que esto implique desarrollarla nuevamente.

Dentro de la problemática de los espacios de trabajo compartidos se han generado nuevas alternativas que buscan implementar espacios de memoria distribuida compartida y persistente. Con este espacio de memoria se busca simplificar las operaciones fundamentales como el intercambio de mensajes y el manejo de sesiones de grupos.

Ejemplos de estos modelos, son los basados en **espacios de tuplas** de objetos, como Linda [Woo99] y Java Spaces [Fre99], donde se separa un servidor genérico y las operaciones asociadas a la comunicación se encapsulan dentro de los objetos. Cada cliente se registra en el espacio, determinando las tuplas con las que interactúa y el paso de mensajes se realiza mediante el tratamiento de los eventos ocurridos sobre dichas tuplas.

La problemática de la colaboración está evolucionando hacia nuevos requerimientos de comunicación y de interacción, gracias al avance alcanzado con la tecnología de la Web. Esta evolución tiene numerosas corrientes, de las cuales se destaca el adoptar la arquitectura de la Web como la base para la construcción de un sistema colaborativo.

La adopción total de la arquitectura Web no es aún suficiente para satisfacer los requerimientos de los sistemas colaborativos. La adopción parcial de esta arquitectura implica el desarrollo de extensiones de las capacidades básicas del servidor HTTP y los métodos definidos en este protocolo, buscando con ello que la Web llegue a convertirse en un ambiente de colaboración completo.

Capítulo 3

Alliance y la edición colaborativa en Internet

Dentro de CSCW, la edición colaborativa se ha consolidado como una disciplina denominada **Edición Colaborativa Asistida por Computadora** (*"CSCWriting"*). Dicha disciplina ha tenido numerosas contribuciones enfocadas al desarrollo de sistemas y herramientas que facilitan la producción de documentos en forma distribuida y colaborativa.

Los desarrollos en CSCWriting surgen como respuesta a una clara necesidad actual: producir documentos de mejor calidad, mediante las contribuciones de diferentes autores, el intercambio de sus experiencias y mejorando la forma como ellos pueden interactuar entre sí.

Esta necesidad establece los requerimientos para los componentes fundamentales de un sistema de edición colaborativa, como lo son entre otros, la administración de documentos, la comunicación de grupos y el acceso a la información a través de Internet. Adicionalmente, en estos sistemas se busca dar apoyo a las diversas etapas del proceso de edición para que el trabajo en conjunto sea más eficiente.

Como se trató en el capítulo 2, los sistemas colaborativos adoptan diversas arquitecturas para su implementación. Estas arquitecturas se enfocan principalmente en los problemas de la comunicación, la coordinación de eventos, la conciencia de grupo y la memoria de grupo. En el contexto de la edición colaborativa, dichas arquitecturas se concentran también en la manipulación y el intercambio de los documentos como entidades fundamentales de información.

La arquitectura de la Web es en la actualidad una arquitectura muy popular, gracias a las facilidades que provee para el acceso a documentos remotos a través de Internet. Sin embargo, los métodos de su protocolo estándar HTTP (GET, PUT, POST, etc.) no ofrecen el soporte necesario para la edición colaborativa distribuida, por lo que resulta más eficiente utilizar otros modelos para el manejo de los documentos. Alliance es un editor colaborativo en Internet, que combina la arquitectura cliente-servidor propia de la Web, con extensiones CGI del servidor HTTP, para el manejo de los documentos en forma distribuida.

Las interfaces de usuario de los editores colaborativos también constituyen otra problemática de investigación. Estas interfaces se caracterizan por la representación de un espacio de trabajo, utilizando el concepto de una o múltiples vistas WYSIWYG del documento sobre el cual se trabaja en forma colaborativa. En dichas interfaces, no sólo se requiere un manejo de la consistencia del documento, sino que se debe proporcionar un alto grado de interacción entre los usuarios.

El presente capítulo aborda la edición como actividad colaborativa y los requerimientos de los sistemas que buscan facilitar este proceso. Asimismo, se analizan las características del editor Alliance y su comparación con otros editores colaborativos en Internet. Finalmente, se establecen los elementos y las herramientas que facilitan la interacción síncrona en un espacio de trabajo compartido, para mejorar la producción de documentos en Alliance.

3.1. Características de la edición colaborativa

La producción de un documento, mediante las contribuciones de diferentes autores, involucra una serie de características relacionadas con la coordinación de las actividades, el papel que puede desempeñar cada autor en un momento dado y el manejo de dichas contribuciones, para que se reflejen en el trabajo de todo el grupo de autores.

Estas características determinan los requerimientos básicos para el diseño de un sistema orientado a facilitar dicha actividad de edición. A continuación, se revisan estas características y los requerimientos funcionales que surgen a partir de ellas.

3.1.1. Fases del proceso de edición colaborativa

La elaboración de un documento es un proceso dinámico en el tiempo que implica la transición entre una serie de fases en las cuales los autores llevan a cabo sus actividades. En [PoBa92][Neu94][Sha93], dichas fases se definen como:

- **Fase de planeación**: corresponde al inicio del proceso de edición. En esta fase los autores establecen los objetivos, el contexto global de los documentos y su estructuración.
- **Fase de escritura**: en esta fase los autores realizan sus contribuciones generando diferentes estados del documento. Durante esta fase, cada autor se entera de las actividades que realizan los demás participantes del proceso.
- **Fase de evaluación**: en esta fase los autores realizan correcciones, comentarios y anotaciones sobre las contribuciones de sus demás colegas.
- **Fase de negociación**: corresponde a un período en el cual los autores discuten sobre las modificaciones que deben hacerse a los documentos. En ocasiones, esta fase se apoya de un proceso de votación para la toma de decisiones.
- **Fase de consolidación**: en esta fase los autores recopilan las estructuras de los documentos para su integración como un todo. Asimismo, los autores revisan los diversos cambios realizados y generan la versión final del documento producido.

Estas fases se llevan a cabo de diversas formas, dependiendo de la madurez del proceso y de la experiencia de sus participantes. Comúnmente, no se presenta un orden secuencial entre dichas fases, sino que éstas pueden repetirse varias veces, en diferentes períodos, hasta que se obtiene la versión definitiva del documento.

En los sistemas de edición colaborativa no sólo se busca apoyar cada fase del proceso de edición, sino también apoyar la transición entre ellas. Lo anterior implica que cada autor debe conocer la información completa sobre los estados en que se encuentra un documento y las actividades que realizan sus demás colegas sobre él.

3.1.2. Modos de interacción

La edición colaborativa involucra diversas formas de colaboración entre los autores. Por una parte, los autores colaboran directamente cuando intercambian ideas o cuando trabajan simultáneamente sobre una sección de un documento. Por otra parte, los autores colaboran indirectamente cuando realizan sus contribuciones en forma individual. Estas situaciones se generalizan en los modos de interacción síncrona y asíncrona.

Interacción síncrona

La interacción de modo síncrono significa que los autores trabajan sobre el mismo documento al mismo tiempo. Esta interacción comprende dos tipos [Bae93][Bae94][Rom98]:

- **Interacción síncrona fuertemente acoplada**: se refiere a que las contribuciones de los autores se integran al documento en tiempo real. Asimismo, los autores perciben simultáneamente todas las acciones realizadas por cada coautor. Generalmente, los editores que cumplen con esta propiedad proveen vistas WYSIWIS (*"What You See is What I See"*)[Ste87] del documento, las cuales son comunes para todos los autores.

- **Interacción síncrona débilmente acoplada**: se refiere a que las contribuciones de los autores se integran al documento con retardos de tiempo variables, que dependen de la voluntad del autor para difundir sus contribuciones y de la decisión de los demás coautores para aceptarlas. A diferencia de la vista común WYSIWIS, la visión del estado global de un documento puede no ser la misma para cada autor, debido al desfase existente entre la difusión y la aceptación de las contribuciones.

La interacción síncrona fuertemente acoplada tiene la ventaja de que los autores mantienen un único estado consistente del documento. Además, los autores pueden confrontar sus ideas en forma directa, por lo que esta interacción se utiliza principalmente en las fases de planeación y negociación. Sin embargo, el potencial de esta interacción se ve reducida cuando se trabaja en sitios ubicados a grandes distancias. De igual forma, este tipo de interacción debe manejarse en forma apropiada, ya que los autores no deben verse afectados o interrumpidos por procesos de actualización constante, como por ejemplo en la transmisión de un texto letra por letra.

La interacción débilmente acoplada tiene la ventaja de permitir a los autores trabajar en modo autónomo o trabajar simultáneamente sobre diversas secciones del documento. Sin embargo, es necesario proveer mecanismos adicionales de sincronización y de control de versiones, para garantizar que los autores mantengan una versión consistente del documento.

Interacción asíncrona

El modo de interacción asíncrona significa que cada autor trabaja sobre un documento en diferentes instantes de tiempo. Esta situación es muy común, ya que la mayor parte del trabajo de edición se lleva a cabo en forma asíncrona, donde las contribuciones se efectúan en forma individual e independientemente de las acciones de los demás coautores.

De acuerdo con [Bae93][PoBa92], este modo de interacción es utilizado principalmente en las fases de escritura y evaluación, puesto que son fases en las que los autores requieren trabajar en cierta forma aislados y sin que los eventos ocurridos interfieran en sus actividades.

Los sistemas de edición colaborativa deben combinar los dos modos de interacción bajo dos aspectos principales [Rom98]:

- permitir tanto la interacción síncrona como la asíncrona, con el fin de que los autores se adapten a diferentes circunstancias, eventos, etc., en diferentes situaciones de interacción entre ellos, y
- el sistema debe permitir la transición entre ambos modos de interacción, para que se ajuste a la flexibilidad requerida en el proceso de edición.

3.1.3. Roles

La edición colaborativa es una actividad que implica la organización y la coordinación del trabajo que realizan los diferentes autores. Los roles que juegan los autores durante la producción de documentos constituyen un elemento fundamental para la coordinación del trabajo conjunto.

En general, un rol permite establecer la relación que existe entre un autor y un objeto de colaboración. A partir de esta relación se pueden controlar los eventos que suceden durante el proceso de edición, como por ejemplo en las escrituras simultáneas sobre la misma parte del documento.

En estudios como [PoBa92][Neu90][Bae93][Sha93], se han identificado los principales roles que toman los autores durante la producción de documentos. Estos roles se definen como sigue:

- **Escritor**: es quien convierte las ideas en texto, registra dicho texto y realiza cambios sobre él.
- **Consultor**: es quien participa activamente en el proceso de edición pero no contribuye en la redacción del documento de manera directa.
- **Editor**: es quien verifica, corrige y prepara el documento para su publicación.
- **Revisor**: es quien proporciona comentarios sobre el texto del documento.

En los sistemas de edición colaborativa los roles han sido incorporados como mecanismos para controlar el acceso sobre los documentos, por ejemplo, para garantizar que sólo un autor, cuyo rol es de escritor o de editor, pueda agregar sus modificaciones en un instante de tiempo determinado. Análogamente, en relación con la coordinación del trabajo, en estos sistemas se han establecido roles de administrador o de coordinador, quienes se encargan principalmente de la distribución y la estructuración del documento, la recopilación de las contribuciones y la asignación de los demás roles.

En contextos reales de producción de documentos, los autores pueden eventualmente cambiar de rol según su experiencia, su conocimiento o su compromiso con el proceso. Por ello, en los sistemas de edición colaborativa es necesario facilitar la asignación de roles durante la fase inicial y permitir que los autores tomen diversos roles durante el resto del proceso.

3.1.4. Conciencia de grupo

El término conciencia, en el contexto de la edición colaborativa, se refiere al entendimiento que un autor tiene de las actividades, las intenciones y la evolución de los documentos de trabajo. Cada autor promueve la conciencia de grupo cuando transmite a sus colegas sus propias acciones individuales y sus contribuciones.

El conocimiento de quien está trabajando sobre un documento, el momento en que se realiza dicho trabajo y las modificaciones efectuadas, son elementos básicos que no sólo benefician la coordinación del trabajo, sino que establecen parámetros para que los autores tomen decisiones sobre las acciones que deben llevarse a cabo.

Con el fin de facilitar la conciencia de grupo los sistemas de edición colaborativa se apoyan en las características de interacción síncrona y asíncrona. Por ejemplo, en los sistemas asíncronos se proveen elementos como comentarios, anotaciones asociadas al documento y elementos gráficos de notificación, que le indican a un autor la ocurrencia de un evento para que decida si se acepta o se rechaza el cambio de estado que este implique.

En los sistemas síncronos la conciencia de grupo se facilita con elementos de notificación inmediata que den la sensación de que se trabaja en tiempo real. Algunos sistemas incorporan telepunteros o barras de estado, que indican la sección del documento sobre la cual un autor realiza sus acciones. En otros sistemas, se proveen registros históricos de las operaciones realizadas por cada autor, los cuales son difundidos entre todos los participantes cada vez que se realiza una acción.

Como ya se ha mencionado, conocer los eventos en el instante en que suceden resulta ser más beneficioso para actividades como la negociación o la discusión. Asimismo, en estudios como [Mit95], se demuestra que para actividades como la modificación del documento, no es necesario que los autores estén trabajando al mismo tiempo para enterarse de los cambios ocurridos. Además, la coordinación de la interacción síncrona entre los autores puede introducir mayor complejidad en el proceso de edición. Un ejemplo de esta situación ocurre cuando se requiere representar en una interface WYSIWIS, la información de un gran número de autores y cada actividad que están realizando.

Por otra parte, los mecanismos de apoyo a la conciencia de grupo síncrona (comunicación directa, expresiones, gestos, etc.) tienen asociado el problema de su implementación en tiempo real, especialmente cuando los autores se ubican en sitios geográficos distantes, utilizando medios de conexión sujetos a fallas como Internet.

Por estas razones, los sistemas de edición colaborativa deben combinar mecanismos de conciencia de grupo, tanto síncronos como asíncronos, para que se ajusten a las necesidades de la producción de documentos en forma distribuida y cooperativa.

3.1.5. Negociación

De acuerdo con [PoBa92][Mit96][Neu94], una de las principales fases en la edición de un documento comprende la fase de negociación. La información que se genera en esta negociación es información fundamental para lograr una versión del documento con la cual todos los autores estén de acuerdo. Esta información debe ser comunicada de manera oportuna entre los respectivos autores, a través de mecanismos confiables y eficientes.

En los sistemas de edición colaborativa, la negociación entre los autores se ha resuelto mediante varias alternativas. Al igual que para la conciencia de grupo, estas alternativas se apoyan en los modos de interacción que caracterizan al editor colaborativo.

En los editores asíncronos, los medios de negociación más comunes son proporcionados por las herramientas de anotaciones de texto[1] y de mensajes de audio o de video, ligados al documento. Los editores síncronos proveen comúnmente herramientas de comunicación como Chats, conferencia de audio y de video.

La negociación es una actividad más efectiva, cuando los autores trabajan al mismo tiempo o mantienen un mecanismo de comunicación directa y constante. Por ello, las herramientas síncronas benefician en mayor grado la actividad de negociación, puesto que son más ajustadas a la necesidad de proveer comunicación en tiempo real.

En un sistema de edición colaborativa, el proceso de edición debe ser percibido como un proceso global de negociación entre los autores, sobre el contenido y el objetivo propio del documento. Por ello, las herramientas de negociación deben contemplar tanto las contribuciones hechas sobre el documento, como la información que se genera en torno a él.

3.1.6. Requerimientos para un sistema de edición colaborativa

Para el diseño y el desarrollo de un sistema de edición colaborativa, es necesario considerar requerimientos adicionales a los que se mencionaron anteriormente. Estos requerimientos reflejan las necesidades y las tendencias actuales de estos sistemas y se resumen como sigue:

[1] Conocidas también como NoteCards

- **Compatibilidad con otros sistemas no colaborativos.** Se refiere a que un autor pueda trabajar sobre sus documentos en forma transparente, en los editores individuales y los sistemas de edición colaborativa. Por ello, es necesario que dichos sistemas soporten los diversos estándares de formatos de documentos, como ASCII, LATEX, TXT, MS Word, PDF, PostScript, HTML, XML, etc.

- **Posibilidad de editar en modo autónomo.** Se refiere a que un sistema debe permitir que los autores trabajen de manera aislada, sin interrupciones, ya que en las fases de escritura y de evaluación, esto es más apropiado.

- **Transición entre el modo de edición autónomo y el modo colaborativo.** Un autor debe poder integrar sus contribuciones hechas en forma individual, de manera que los demás autores puedan enterarse del estado del documento y continuar con el trabajo conjunto sobre él.

- **Desempeño y escalabilidad.** Un sistema debe proveer tiempos de respuesta cortos entre los eventos que suceden y la percepción de los mismos. Asimismo, un sistema no debe verse afectado por el número de autores que participen en el proceso de edición, aún si se encuentran trabajando en sitios geográficos distantes.

- **Tolerancia a fallas.** Debido a que los medios de comunicación entre grandes distancias son vulnerables, un sistema debe ser suficientemente robusto para garantizar la consistencia del documento, por ejemplo, cuando ocurren desconexiones temporales de los usuarios o cuando suceden fallas en los equipos interconectados.

- **Interface del ambiente de colaboración.** Se refiere a la forma como se visualiza y se presenta un documento para que sea coherente para todos los autores. En la interface se incluyen elementos como vistas WYSIWIS, identidad de los autores, elementos gráficos de información (iconos) y vistas WYSIWYG. En estas últimas, la información visualizada del documento debe ser exactamente la información que se obtiene al imprimirlo.

- **Control de versiones.** Se refiere al manejo de las versiones de un documento, lo cual es una característica aplicada a cualquier fase del proceso de edición. Un sistema de edición colaborativa debe permitir a los autores diferenciar y recuperar diferentes versiones del documento, así como poder compararlas y generar a partir de ellas una nueva versión.

3.2. Arquitecturas para el almacenamiento de documentos

En el desarrollo de sistemas de edición colaborativa se han adoptado las diversas arquitecturas de sistemas distribuidos para el almacenamiento y la manipulación de los documentos [Dec99][Cou95]. Estas arquitecturas utilizan Internet como el medio para la interconexión de sitios geográficos distantes, mediante los protocolos de TCP/IP.

En estas arquitecturas se definen dos componentes básicos: el cliente y el servidor. En el contexto de los sistemas de edición colaborativa, el cliente corresponde al conjunto de procesos que facilitan todas las operaciones relacionadas con la edición de documentos. De igual forma, el servidor corresponde al conjunto de procesos necesarios para la comunicación entre los clientes, la coordinación de eventos, el control del acceso y la consistencia de los documentos.

Cada arquitectura de almacenamiento de documentos establece diferentes ubicaciones (sitios) para los clientes y los servidores, definiendo las operaciones para el acceso y la manipulación de los documentos. A continuación se revisan las características de estas arquitecturas.

3.2.1. Centralizada

Los documentos residen en un sitio servidor y los sitios clientes realizan peticiones al servidor para la edición del documento. Esta edición se realiza directamente sobre el documento que reside en el servidor, el cual utiliza un proceso para la coordinación y el control de las modificaciones que realizan los clientes.

Esta arquitectura ha sido ampliamente utilizada en los sistemas multiusuario tradicionales. Sin embargo, es una arquitectura que resulta ser muy limitada respecto a las necesidades de escalabilidad y tolerancia a fallas de los editores colaborativos actuales:

- Escalabilidad: debido a que existe un único servidor central de gran capacidad de procesamiento y almacenamiento, a medida que crece el número de usuarios que interactúan con el sistema el incremento en la carga del servidor degrada notablemente su desempeño.

- Tolerancia a fallas: debido a que el almacenamiento de los documentos y los procesos de coordinación residen sólo en un servidor, una situación de falla en dicho servidor genera la inoperabilidad total del sistema.

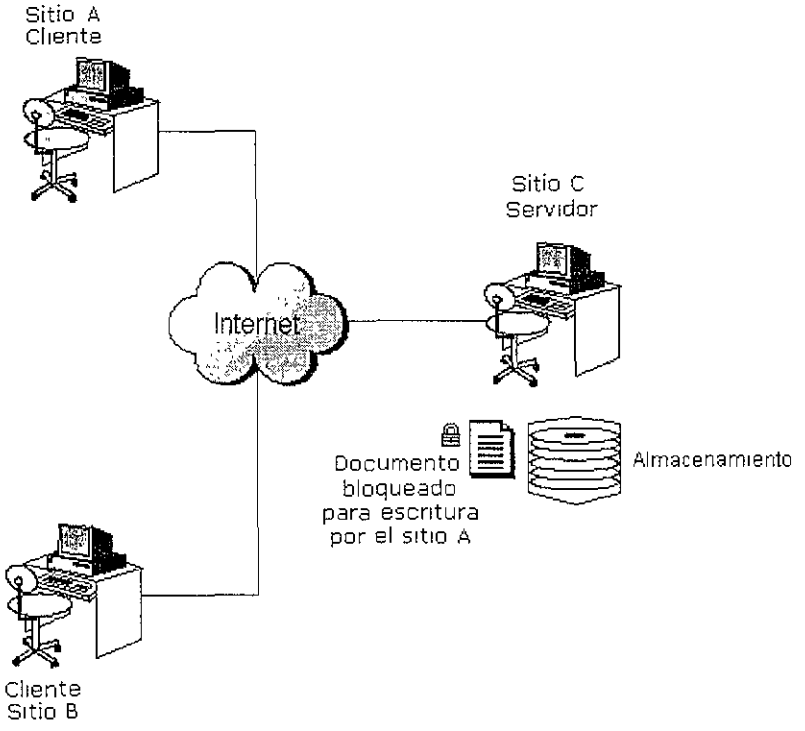


Figura 3.1. Arquitectura de almacenamiento centralizado.

La figura 3.1 muestra dos sitios clientes A y B, que trabajan sobre el documento almacenado en el sitio servidor. Los clientes no almacenan en forma local los documentos, sino que utilizan copias en memoria para trabajar con ellos. El servidor utiliza un mecanismo de bloqueo para controlar las modificaciones concurrentes. En la figura, el sitio A bloquea el documento para escritura y el sitio B sólo tiene acceso de lectura mientras no se libere dicho bloqueo.

Esta arquitectura es muy poco utilizada en los editores colaborativos actuales, pero constituye una base fundamental en el diseño de arquitecturas más funcionales como la híbrida, la cual se trata más adelante en la sección 3.2.3.

3.2.2. Distribuida

En esta arquitectura, una copia (réplica) de cada documento es distribuida entre cada autor que participa en el proceso de edición. De esta forma, cada cliente trabaja sobre su copia del documento de manera aislada e independiente.

En cada cliente existe un conjunto de procesos para la distribución de las copias de los documentos y el tratamiento de su consistencia: cada autor puede modificar su copia local en diferentes instantes de tiempo y el sistema debe garantizar que el documento sea concebido como un todo, de forma coherente para todos los autores. Estas características sugieren que en cada sitio cliente exista también un servidor de documentos, ya que cada autor puede crearlos y distribuirlos para que sean tratados en forma colaborativa.

Para el tratamiento de la consistencia de los documentos copiados en cada sitio se utilizan dos técnicas principales: el manejo de copias maestras y esclavas y el manejo de versiones del documento.

La primera técnica consiste en que el documento distribuido se compone de una copia maestra para las operaciones de escritura y de múltiples copias esclavas para las operaciones de lectura. La copia maestra del documento migra entre los diversos sitios con el fin de habilitar la escritura en ellos: sólo el sitio que posea la copia maestra puede realizar operaciones de escritura. De esta forma, el documento permanece consistente ya que sólo un autor puede modificarlo en un momento dado y los cambios realizados son reflejados en las diferentes copias de sólo lectura de los demás autores.

La segunda técnica de consistencia se refiere a que en cada sitio se crean diferentes versiones de un mismo documento, a partir de sus copias distribuidas. La consistencia del documento se logra mediante la reunión de todas las versiones producidas y su comparación para determinar los cambio definitivos. Para ello, se utiliza un mecanismo de negociación entre los sitios, con el fin de resolver conflictos sobre los cambios realizados. Estos mecanismos varían dependiendo del nivel donde se aplique la comparación de las versiones, es decir, si la comparación se realiza sobre la estructura del documento, sobre cada párrafo, sobre cada línea, sobre cada carácter, etc.

La arquitectura distribuida es muy apropiada para sistemas escalables y tolerantes a fallas. El desempeño del sistema no se ve afectado cuando el número de usuarios crece, puesto que cada uno trabaja sobre su copia local del documento. Asimismo, cuando existe una falla en algún sitio, ésta no afecta el resto del sistema, puesto que cada uno trabaja en forma independiente.

La desventaja de esta arquitectura radica en que es necesario emplear un proceso de manejo de la consistencia del documento, el cual agrega mayor complejidad a las funciones de cada sitio. Además, dicho control de la consistencia genera una mayor cantidad de mensajes entre cada sitio, lo cual agrega mayor tráfico en los canales de comunicación.

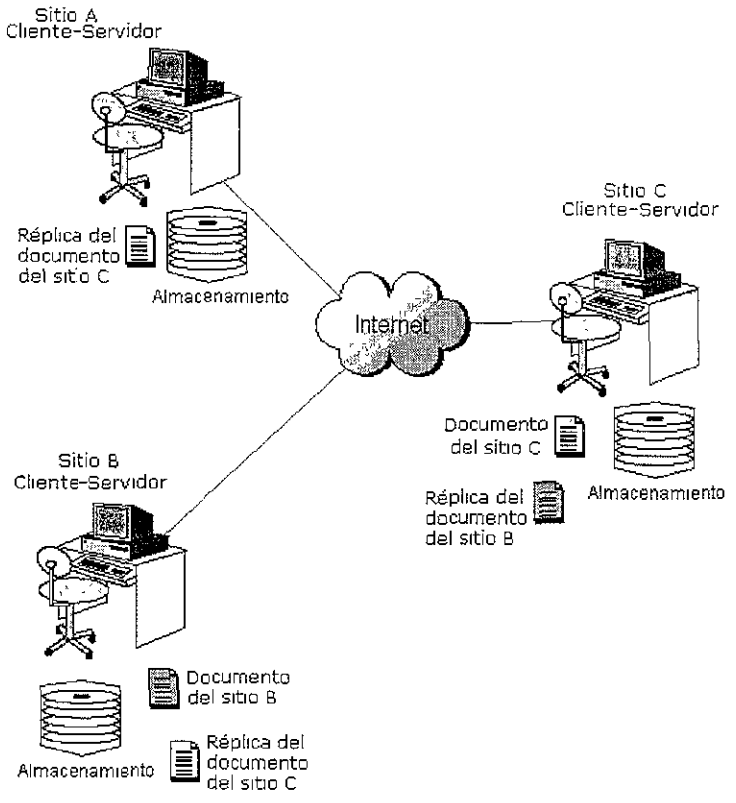


Figura 3.2. Arquitectura de almacenamiento distribuido.

En la figura 3.2 se muestra la distribución de documentos entre los sitios cliente-servidor A, B y C. El documento del sitio C, es copiado hacia los sitios A y B, mientras que el documento del sitio B es copiado solamente hacia el sitio C. Además de copiar los documentos, es necesario que cada sitio difunda todos los cambios que se hagan sobre ellos.

3.2.3. Híbrida

Esta arquitectura combina las características de las arquitecturas centralizada y distribuida, por lo que es la alternativa más usada en los editores colaborativos a gran escala. En general, en cada ubicación existen procesos clientes y servidores al igual que en la arquitectura distribuida, pero las copias de los documentos se tratan de forma diferente.

La combinación de las arquitecturas centralizada y distribuida puede darse en múltiples formas. La combinación más común es la utilizada para la publicación de documentos en la Web: los documentos se almacenan en forma centralizada en los sitios servidores y los sitios clientes obtienen una copia del documento (método *HTTP-GET*) para realizar las modificaciones en forma local. Finalmente, utilizando extensiones del servidor de Web, mediante CGIs o modificaciones del método *HTTP-PUT*, los documentos son regresados de nuevo al servidor para su publicación.

El principal ejemplo de la modificación del método *HTTP-PUT*, para manipular documentos HTML, es el que propone el grupo de investigación WEBDAV del IETF[2] (*"Internet Engineering Task Force"*), con el protocolo WEBDAV [Whi97][Sle97]. En este protocolo, se modifica el método *HTTP-PUT* estándar, con funciones para transmitir los cambios que realizan los clientes sobre las copias locales de los documentos. Cuando ocurre dicho cambio, sólo se transmite hacia el servidor la información suficiente para generar el nuevo estado del documento. Con estas funciones, el método *HTTP-PUT* permite un control de versiones para cada documento del servidor que es copiado en los clientes.

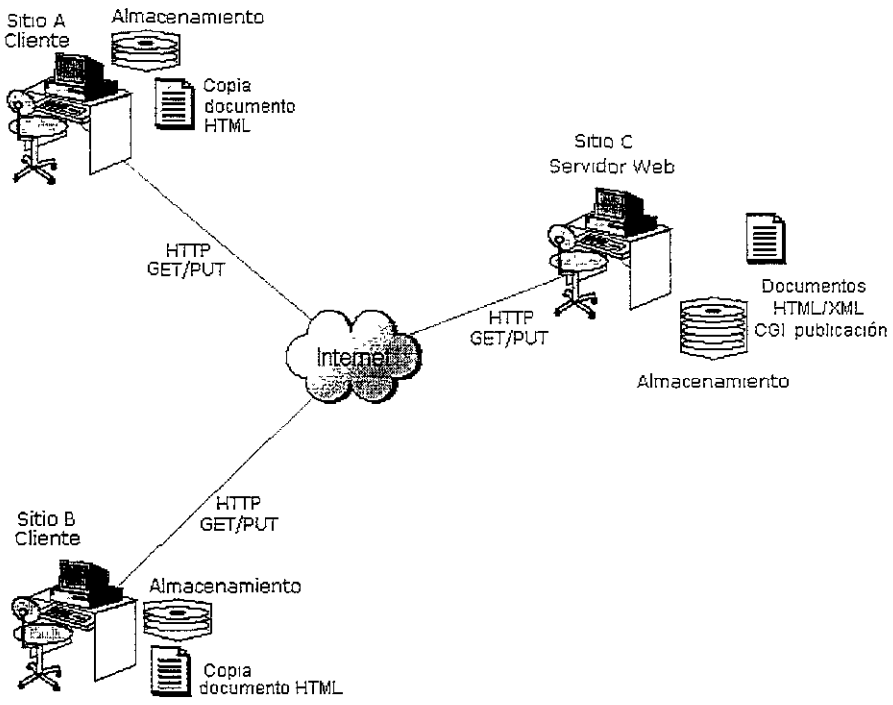


Figura 3.3. Arquitectura de almacenamiento de la Web.

En la figura 3.3 se presenta la arquitectura básica de almacenamiento y publicación de documentos en la Web. Los sitios clientes A y B obtienen copias de los documentos almacenados en el sitio servidor Web, mediante el método *HTTP-GET*. Las modificaciones locales efectuadas al documento son enviadas al servidor mediante *HTTP-PUT* o ejecutando en el servidor el CGI **publicación**. Estas extensiones son las responsables de mantener la consistencia del documento.

La arquitectura híbrida de la Web es relativamente fácil de implementar pero tiene la desventaja de verse afectada por los retardos y desconexiones que ocurren en Internet. Más aún, bajo esta arquitectura resulta ser muy complejo el implementar mecanismos de tolerancia a fallas, debido a las limitaciones del protocolo HTTP (ver sección 2.3.3) y a que se mantiene un servidor de documentos centralizado.

[2] IETF http://www.ietf.org.

Esta situación ha dado origen a extensiones de la arquitectura híbrida. En dichas extensiones, los documentos y sus copias residen solamente en algunos de los sitios cliente-servidor conectados a través de Internet. Los demás sitios accesan los documentos a través de un ambiente de conexión confiable, generalmente una red de área local. Cada cliente realiza sus modificaciones directamente en el servidor de documentos, para que éste se encargue de difundirlas a los demás.

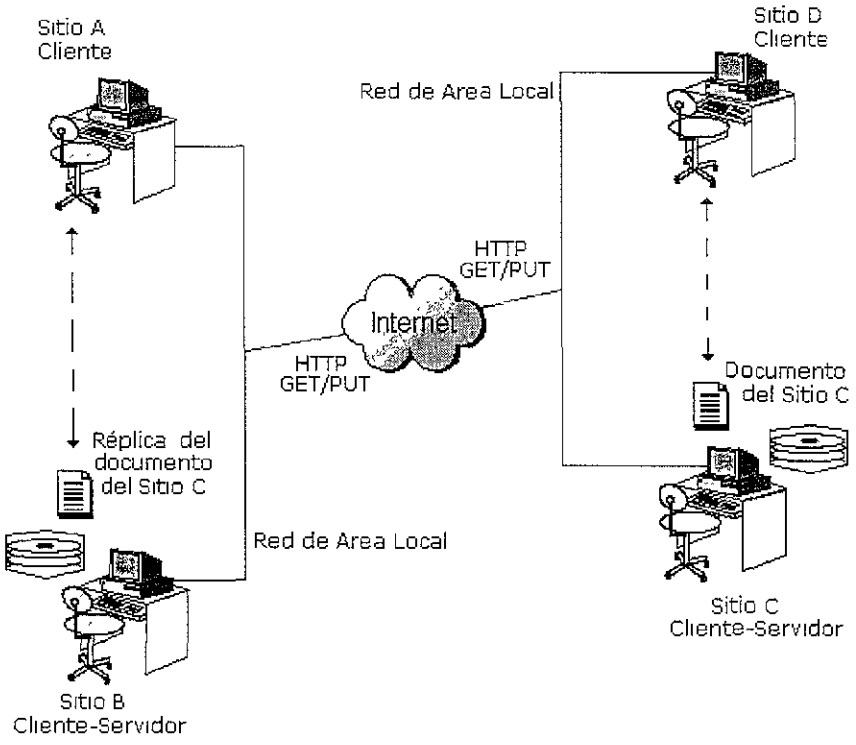


Figura 3.4. Arquitectura híbrida extendida.

En la figura 3.4 se muestra como en la arquitectura híbrida extendida, los documentos se copian solamente en los sitios cliente-servidor B y C. Los sitios cliente A y D accesan en forma centralizada el documento, directamente desde sus servidores de documentos B y C, ubicados en sus respectivas redes de área local. Los eventos que ocurren en cada sitio son difundidos a los demás mediante los servidores de documentos. De esta forma, los cambios en el documento realizados por el sitio A, se almacenan inicialmente en la copia ubicada en B y posteriormente el documento es actualizado en el sitio original C, permitiendo que C y D continúen su trabajo sobre la versión actualizada del documento.

Una extensión especial de la arquitectura híbrida de la Web, es la basada en el modelo middleware de tres capas (cliente-broker-servidor). En esta arquitectura se utiliza un almacenamiento centralizado, separando dos tipos de sitios servidores: para la atención de peticiones, que generalmente es el servidor de los procesos del sistema (servidor de aplicaciones) y para el control y el almacenamiento de los documentos en una base de datos (servidor de base de datos).

En esta arquitectura, los sitios también se diferencian como clientes y servidores, pero existen varios servidores del sistema, en los cuales se distribuyen los servicios para la atención a los clientes y para la comunicación con el servidor de base de datos. De esta manera, cuando uno

de los servidores del sistema falla, los clientes pueden ser direccionados a los demás servidores para que el sistema continúe operando.

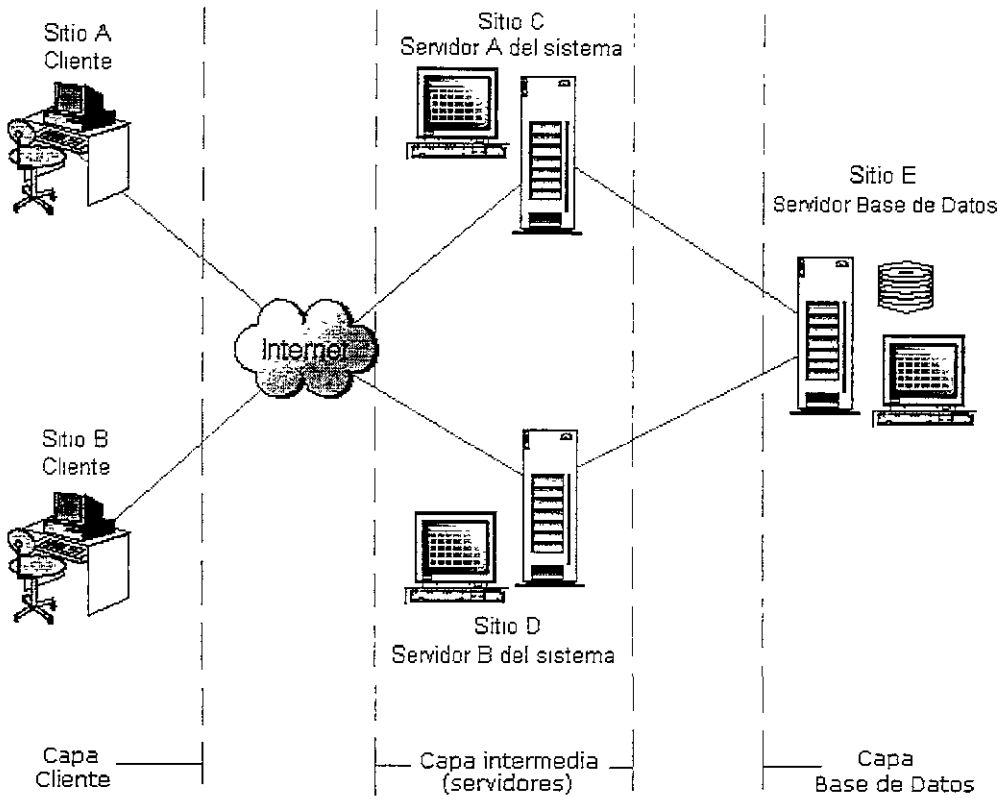


Figura 3.5. Arquitectura de 3 capas.

La figura 3.5 muestra una arquitectura típica de tres capas. Los sitios cliente A y B realizan peticiones de documentos a los sitios servidores del sistema C y D. Nótese que en caso de una falla en alguno de los servidores el sistema puede seguir operando. Adicionalmente, el servidor de base de datos (sitio E) se encarga de las operaciones de control y bloqueo de los documentos almacenados, disminuyendo la carga en los servidores del sistema.

La desventaja de esta arquitectura es que implica un mayor costo de implementación y los procesos de los servidores son más complejos. Además, se requieren procesos adicionales para la comunicación entre los servidores del sistema y de bases de datos. Sin embargo, este modelo es muy utilizado en sistemas distribuidos a gran escala, donde el factor de la disponibilidad y la tolerancia a fallas es muy crítico.

3.3. Características de Alliance

El editor colaborativo Alliance [Rom98] permite que diversos autores, ubicados en sitios geográficos diferentes, puedan trabajar conjuntamente sobre documentos estructurados. Este editor se concentra principalmente en los requerimientos desde el punto de vista de los autores, con el fin de facilitar el manejo de los documentos en un ambiente de edición colaborativa y distribuida.

Alliance utiliza una arquitectura distribuida para la administración y el almacenamiento de los documentos de trabajo, mediante el manejo de copias de los documentos y los fragmentos, en cada sitio que participa en el proceso de edición.

Por otro lado, Alliance adopta parcialmente la arquitectura de la Web para su funcionamiento, explotando ampliamente el modelo cliente-servidor y los métodos GET y POST del protocolo HTTP. Sin embargo, Alliance no utiliza la interface propia de la Web (un navegador convencional), ni manipula documentos de formato HTML. Alliance implementa su propia interface, basada en el editor THOT[3] y maneja los documentos codificados en un formato específico denominado pivote (extensión .PIV).

A continuación se revisan las principales características de la edición colaborativa en Alliance, los mecanismos de apoyo a la conciencia de grupo y la arquitectura sobre la cual está basada su implementación.

3.3.1. Principios de la edición colaborativa en Alliance

El sistema Alliance tiene como objetivo principal facilitar la manipulación de documentos compartidos, mediante su estructuración en entidades básicas denominadas fragmentos. Esto se hace basándose en el principio de que en la edición de documentos, por un grupo de autores, cada uno juega un papel bien definido (rol) sobre los diferentes fragmentos del documento. Alliance hace énfasis en la importancia del autor como principal participante del proceso de edición de un documento, por lo que el manejo de roles flexibles es una característica clave.

La edición colaborativa en Alliance se basa en tres principios fundamentales: manejo de roles de edición, estructuración de los documentos y apoyo a la conciencia de grupo. A continuación se describen estos principios.

3.3.1.1. Manejo de roles de edición

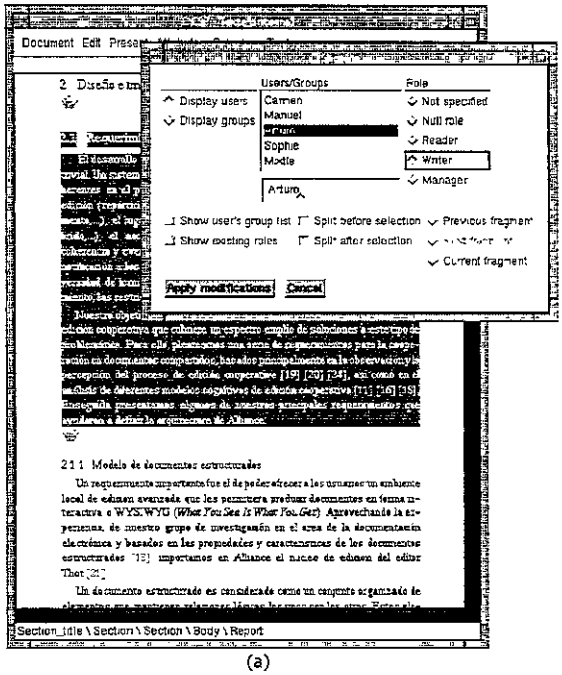
Consiste en asignar a cada autor o grupo de autores, las diferentes operaciones que pueden realizar durante el proceso de edición. Mediante el manejo de los roles, Alliance controla la coherencia y la coordinación del acceso a los fragmentos de los documentos.

Los roles que establece Alliance para los autores son:

- **Escritor**: el cual permite que un autor modifique un fragmento a nivel de su contenido y su estructura.
- **Lector**: el cual permite únicamente la consulta del fragmento.
- **Nulo**: el cual prohibe la visualización de un fragmento que se considere como confidencial.
- **Administrador**: el cual permite la división de los documentos en fragmentos, el establecimiento de los diversos roles sobre ellos y su modificación. Este rol es atribuido al autor que se encarga de la creación del documento inicial.

Cada rol tiene un icono representativo que se visualiza en la parte superior de cada fragmento compartido por los diversos autores. Estos iconos determinan las operaciones atribuidas a cada rol sobre cada fragmento.

[3] Editor de documentos estructurados Thot, http://www.inrialpes.fr/opera/Thot.en.html.



(a)

Roles	Iconos
Escritor	
Lector	
Nulo	
Administrador	

(b)

Figura 3.6. Ejemplo de la asignación de roles sobre fragmentos. (a) Asignación de rol de escritor a un usuario específico. (b) Iconos representativos de los diferentes roles. Fuente: Traducido de [Rom98].

La figura 3.6 ejemplifica la asignación de los roles sobre un fragmento específico. En la parte (a) se muestra la asignación del rol de escritor al usuario **Arturo** sobre el fragmento 2.1. (recuadro seleccionado). En la parte (b) se muestran los diferentes iconos que representan cada rol.

Los roles son diferenciados en dos clases: **roles potenciales** y **roles efectivos**. Los roles potenciales son los establecidos por el administrador inicialmente, los cuales determinan las posibilidades de cooperación de los autores sobre un documento. Los roles efectivos son los que desempeñan los autores en un momento dado en función de la concurrencia en el trabajo de edición.

El rol efectivo de un autor está determinado a partir de su rol potencial y de la actividad de los demás autores sobre el mismo fragmento. Por ejemplo, si dos autores tienen roles potenciales de escritor en un fragmento, sólo uno tiene rol efectivo de escritor en un momento dado, mientras que el otro asume un rol de lector hasta que el primer escritor termine su contribución.

Esta situación conlleva a que los autores cedan su rol de escritor a otros autores, para que puedan realizar las modificaciones del documento. De esta manera, Alliance establece una política de múltiples lectores y un solo escritor, con la cual se garantiza la consistencia de los fragmentos que son modificados por el grupo de autores.

La visualización de dichas modificaciones se realiza mediante un mecanismo de interacción síncrona débilmente acoplada, el cual es controlado por la acción voluntaria que cada autor lleva a cabo cuando desea actualizar su vista del documento.

3.3.1.2. Estructuración de documentos

Consiste en la división de un documento en unidades básicas conocidas como fragmentos, la relación que existe entre ellos y su distribución entre los autores correspondientes. El proceso de división del documento está determinado por el proceso de asignación de roles.

Cada autor que colabora en el proceso de edición posee una vista WYSIWYG personal conformada por los fragmentos que le son asignados por el administrador en el momento de la creación del documento. El documento es manipulado como una estructura de árbol de abstracción, conformada por los fragmentos y sus relaciones entre sí. Esta estructura es utilizada para reconstruir cada vista del documento en cada sitio donde colabora cada autor.

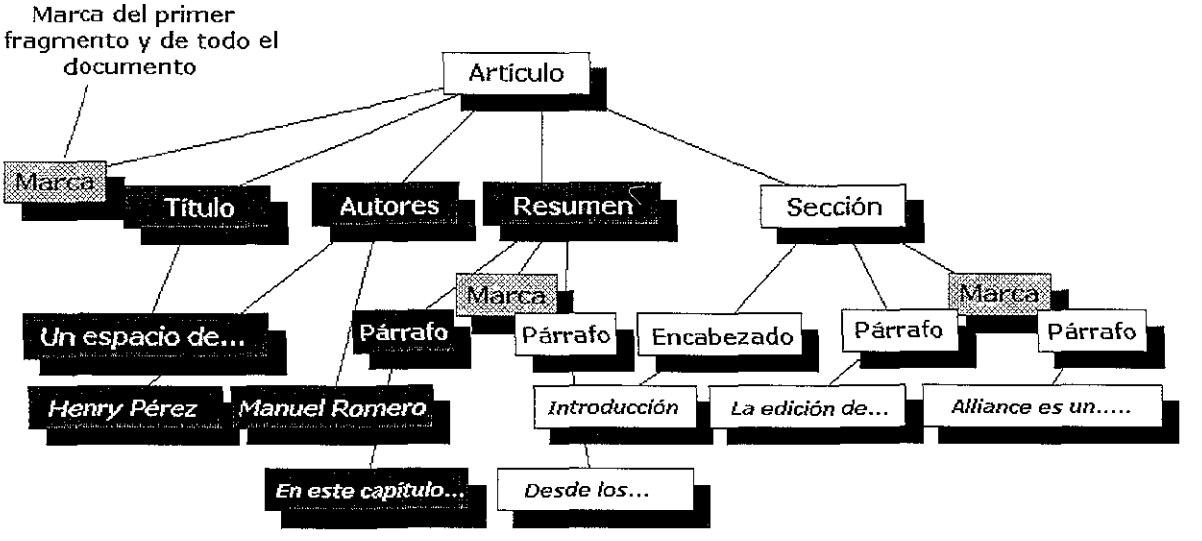


Figura 3.7. Fragmentación del árbol de abstracción de un documento. Fuente: Traducido de [Rom98].

La figura 3.7 presenta un ejemplo de la fragmentación del árbol de abstracción de un documento dividido en tres fragmentos. Cada fragmento está delimitado mediante marcas (transparentes para los usuarios), las cuales sirven para identificarlos de manera única y especificar el rol de edición que se aplica en cada uno. En el árbol de abstracción, cada fragmento comprende las ramas ubicadas a la derecha de cada marca, hasta encontrar la siguiente marca. En la figura, el primer fragmento (recuadros más oscuros) contiene las ramas completas de **Título**, **Autores** y la rama del primer **Párrafo** de la rama **Resumen**. La marca ubicada a la izquierda de la rama **Título** es a su vez la marca que identifica todo el documento.

El administrador, durante la creación del documento, establece los fragmentos en los que se divide un documento y asigna los roles correspondientes para los autores. Por ejemplo, en la figura 3.6 (a) se muestra como al asignar un rol, se establece también un fragmento del documento.

Para determinar los roles que fueron asignados a los autores, Alliance emplea una lista de acceso que puede ser modificada dinámicamente. Esta lista hace parte de la descripción de cada fragmento y almacena los grupos de autores y sus correspondientes roles.

Durante el trabajo de edición, cada autor trabaja sobre los fragmentos a los que tiene acceso, visualizándolos como un documento continuo. Asimismo, los autores pueden visualizar diferentes iconos sobre los fragmentos, puesto que cada uno puede desempeñar diferentes roles en el proceso de edición.

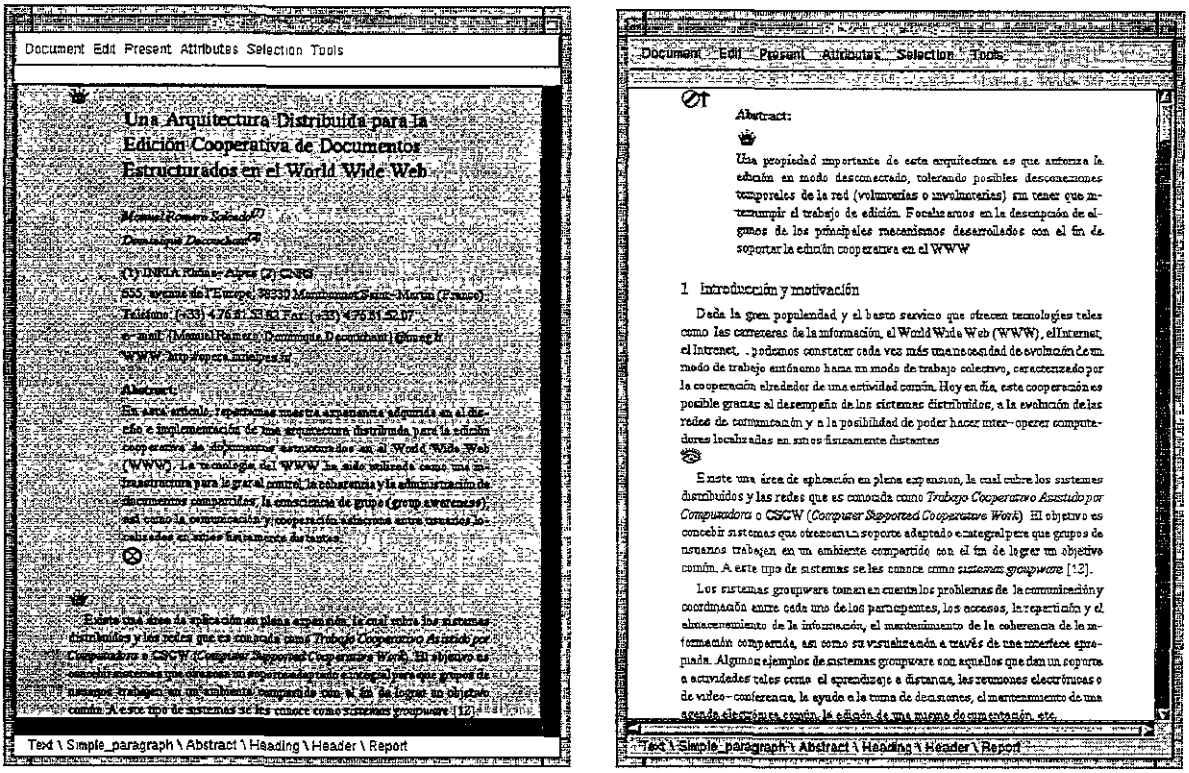


Figura 3.8. Ejemplo de las vistas de los autores 1 y 2 de un documento compartido en Alliance. Fuente: [Rom98].

La figura 3.8 presenta un ejemplo de dos vistas de un documento compartido en Alliance, con los iconos de rol sobre tres fragmentos. El primer fragmento comprende el encabezado del documento (el título y la información de los autores) y el primer párrafo de la sección **Abstract**. El segundo fragmento comprende el segundo párrafo de la sección Abstract y el primer párrafo de la sección **Introducción y motivación**. El tercer fragmento comprende el segundo párrafo de la sección Introducción y motivación.

El **Autor 1** es el administrador del primer y el tercer fragmento mientras que el **Autor 2** es el administrador del segundo fragmento. A su vez, el autor 2 sólo tiene privilegios de lectura sobre el tercer fragmento. Nótese que en la vista del autor 1, no se visualiza el segundo párrafo del Abstract ni el primer párrafo de la sección Introducción y motivación, lo cual se indica con el icono de nulo. Nótese además que en la vista del autor 2, el encabezado del documento tampoco se visualiza, lo cual se indica con otro icono de rol nulo.

En esta figura se presenta una situación donde se ceden los roles: el icono de rol nulo de la vista del autor 2 tiene asociada una flecha hacia arriba. Esto indica que el autor 2 tiene momentáneamente el rol de nulo sobre el primer fragmento, pues éste está siendo editado por el autor 1. Una vez que el autor 1 cede su rol, el autor 2 podrá visualizar el fragmento, recuperando de nuevo su antiguo rol. En la siguiente sección se retoma este concepto de cambio de roles.

3.3.1.3. Apoyo a la conciencia de grupo

La conciencia de grupo permite que cada autor trabaje sobre su propio ambiente particular, percibiendo la evolución de las contribuciones de los demás coautores y controlando la difusión de su propia contribución.

Alliance está basado en el tipo de conciencia de grupo asíncrona [RoDe97][Rom01], donde existe un desfase entre la acción de edición y la percepción por los demás autores. Cada autor percibe las modificaciones de los demás, con un retardo de tiempo variable que depende de la voluntad de un autor en difundir su contribución y de los coautores en aceptarlas. De esta manera, cada autor decide cuando obtener y desplegar en su pantalla la versión más actualizada de los fragmentos que están siendo editados en forma conjunta.

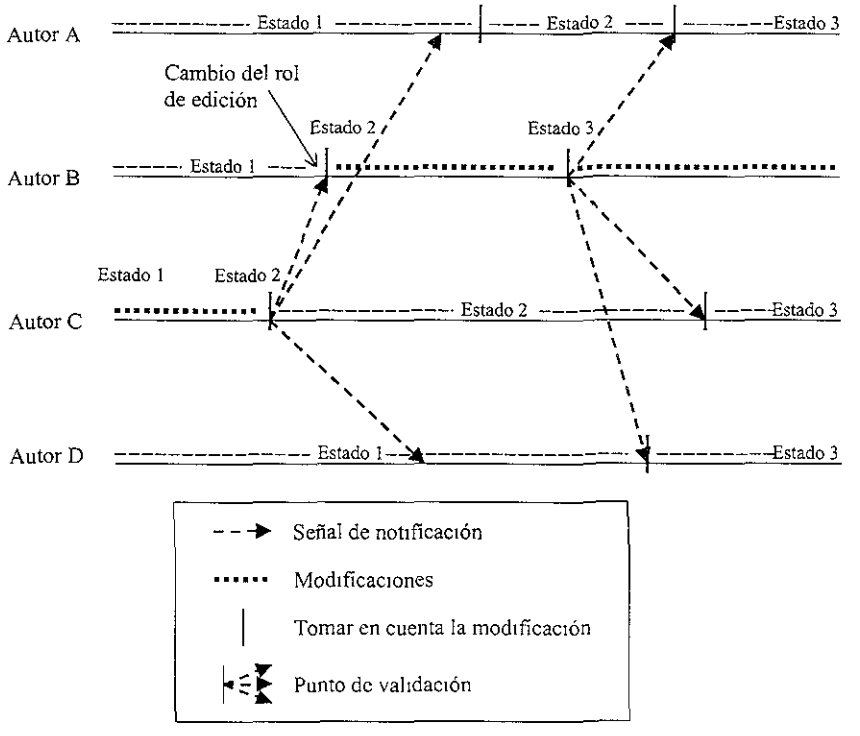


Figura 3.9. Conciencia de grupo asíncrona: acciones de edición y de percepción. Fuente: Traducido de [Rom98].

En la figura 3.9 se presenta el mecanismo de conciencia de grupo asíncrona de 4 autores (A, B, C y D) que trabajan sobre el mismo fragmento en un período de tiempo determinado. Los estados se refieren a las diferentes versiones del fragmento que cada autor visualiza en un instante de tiempo específico. Inicialmente, el autor C (rol de administrador) realiza modificaciones sobre el fragmento.

Después de un tiempo, el autor C decide difundir (validar) sus modificaciones para que sean accesibles a los demás autores y atribuye el rol de escritor al autor B. El autor B cambia su rol a escritor recibiendo la versión del fragmento actualizada por C (estado 2). Los autores A y D solamente reciben la notificación de C y deciden no actualizar las modificaciones en sus vistas del fragmento. Posteriormente, el autor B valida sus modificaciones y los autores A, C y D deciden tomarlas en cuenta para actualizar sus vistas del fragmento, implicando con ello el cambio al estado 3 en cada uno. Nótese que el autor D nunca recibe la versión del estado 2 del fragmento, debido a que se encontraba trabajando en modo desconectado o realizando

otra actividad. Por el contrario, el autor A sí recibe dicha versión porque decide tomar en cuenta las modificaciones antes de que el autor B valide sus cambios.

La figura 3.9 también muestra como se conserva la consistencia de un fragmento: sólo un autor puede modificarlo en un momento dado y los demás autores obtienen la versión actualizada del mismo, en el momento en que deciden tomar en cuenta los cambios validados.

El apoyo a la conciencia de grupo se basa en la extensión del sistema de iconos representativos de los roles de los autores (ver figura 3.6), con las variantes que indican el estado en que se encuentra cada fragmento. Cuando un autor realiza una validación, sus colegas perciben este evento mediante los cambios en los iconos de los fragmentos. Dichos iconos son elementos activos dentro de la interface del documento: al hacer doble *click* en ellos el autor puede cambiar el estado del fragmento al que están asociados. De igual forma, al hacer un click sobre el icono, el autor puede establecer la variante que va a utilizar.

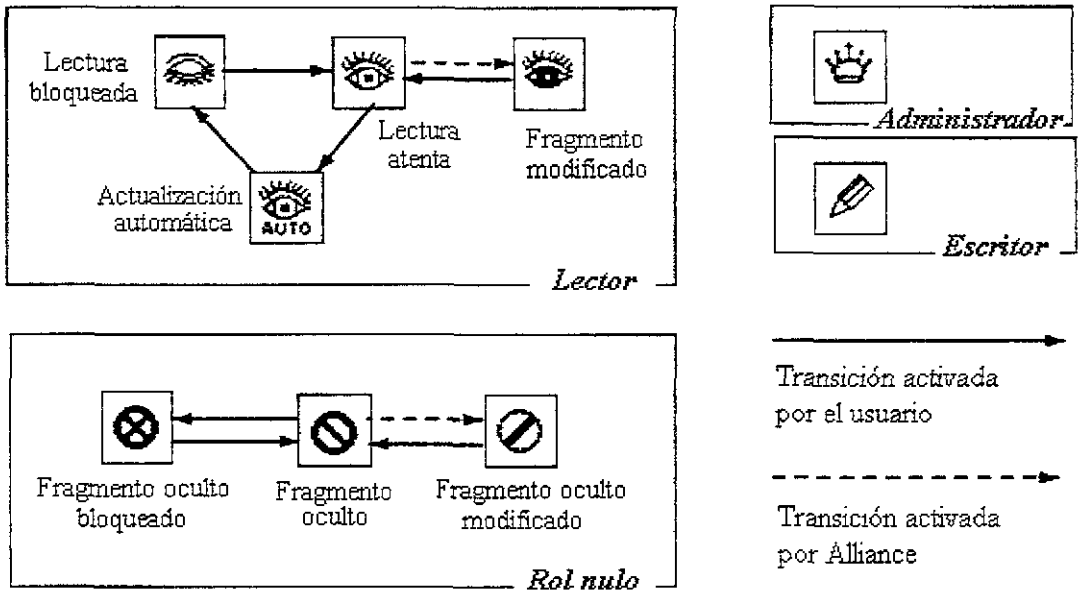


Figura 3.10. Esquema de transición de estados para los iconos de los roles de edición. Fuente: Traducido de [Rom98].

La figura 3.10 presenta el esquema de transición de estados para los iconos representativos de los roles de edición. Cuando un autor tiene el rol de lector sobre un fragmento, visualiza el icono de lectura atenta en su interface. Cada vez que el autor escritor de dicho fragmento haga disponible una nueva versión, el icono del lector cambiará a fragmento modificado. Una vez que el lector obtiene la nueva versión del fragmento (haciendo doble *click* en el icono) el icono del lector regresa nuevamente a lectura atenta.

El icono de lectura bloqueada indica que el lector no desea enterarse de las nuevas versiones del fragmento. El icono de actualización automática indica que el lector permite la actualización del fragmento en el momento en que el escritor difunde una nueva versión.

Para el caso del rol nulo, las variantes del icono de fragmento oculto, fragmento oculto modificado y fragmento oculto bloqueado, se comportan de forma similar que las variantes del icono de lectura. El rol nulo puede ser establecido por el autor administrador o por cada autor que desea bajar su rol sobre un fragmento, para concentrarse en otros fragmentos del

documento. Cuando el autor baja de rol, el respectivo icono del rol es visualizado en la interface con una flecha hacia arriba, lo cual indica que el autor puede volver a tomar su rol en el momento que lo requiera.

Los roles de escritor y administrador no presentan variaciones en los iconos puesto que un fragmento es modificado solamente por un rol efectivo a la vez. Cuando esto sucede, los demás autores sólo pueden asumir el rol de lector o el rol nulo sobre un fragmento, así posean roles potenciales de escritor o de administrador.

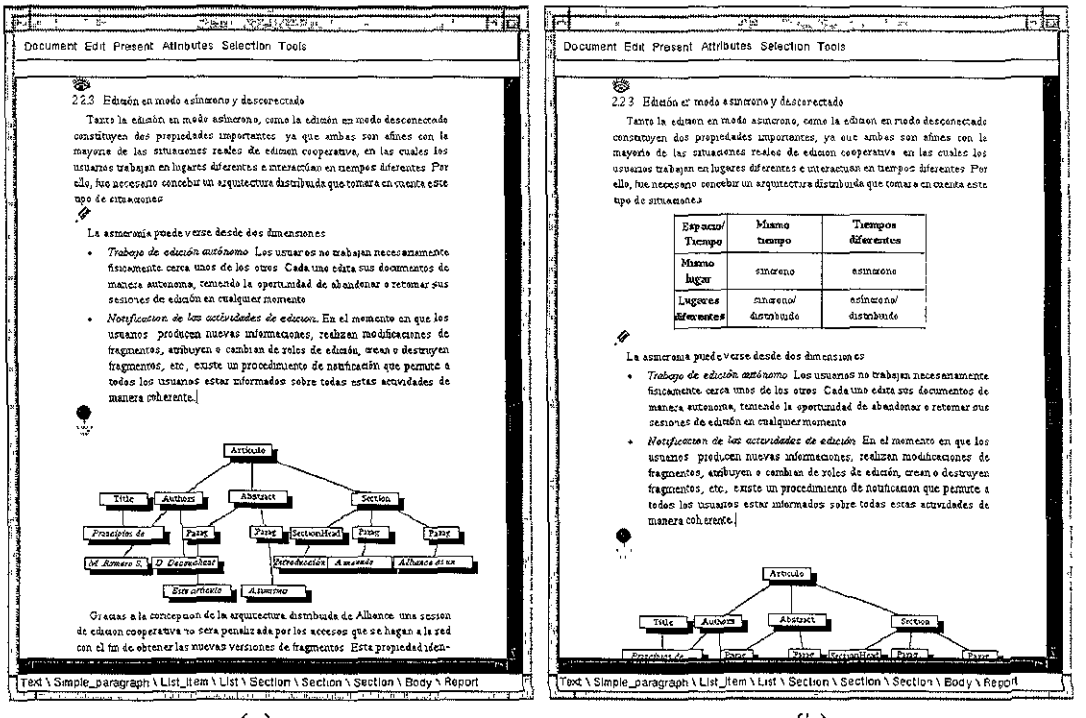


Figura 3.11. Visualización de los cambios de estado de un fragmento.
(a) Fragmento 2.2.3 con icono de lector indicando una modificación.
(b) Mismo fragmento con icono de lector después de visualizar la modificación.
Fuente: [Rom98].

Finalmente, en la figura 3.11 se ejemplifica la visualización de los cambios de estado del fragmento de la sección 2.2.3. de un documento en particular. En la parte (a) se visualiza el icono de lectura del fragmento, con la variante de fragmento modificado, el cual indica que el autor escritor ha validado su modificación. En la parte (b) se muestra el nuevo estado del fragmento con la adición de una tabla y el cambio en el icono de lectura sobre el mismo.

3.3.2. Arquitectura distribuida

Alliance ha sido diseñado con una arquitectura para la Web en la cual se definen tres componentes básicos [RoDe97]:

- **El editor local**: el cual desempeña todas las funciones de edición local para aquellos autores que esperan una realimentación inmediata de su trabajo y no dependen de recursos remotos como la fragmentación, la asignación de roles, la definición de autores y grupos, etc.

- **El asistente**: el cual ejecuta todas las funciones de comunicación a través de la red, las cuales involucran el manejo de retardos en las transmisiones, bloqueo de acceso a documentos remotos, actualización de documentos, envío de eventos, etc.

- **El servidor de documentos**: el cual ejecuta todos los servicios cooperativos de administración de documentos, incluyendo el almacenamiento distribuido de documentos, la migración de las copias, el control de la concurrencia, el bloqueo y desbloqueo de fragmentos, etc.

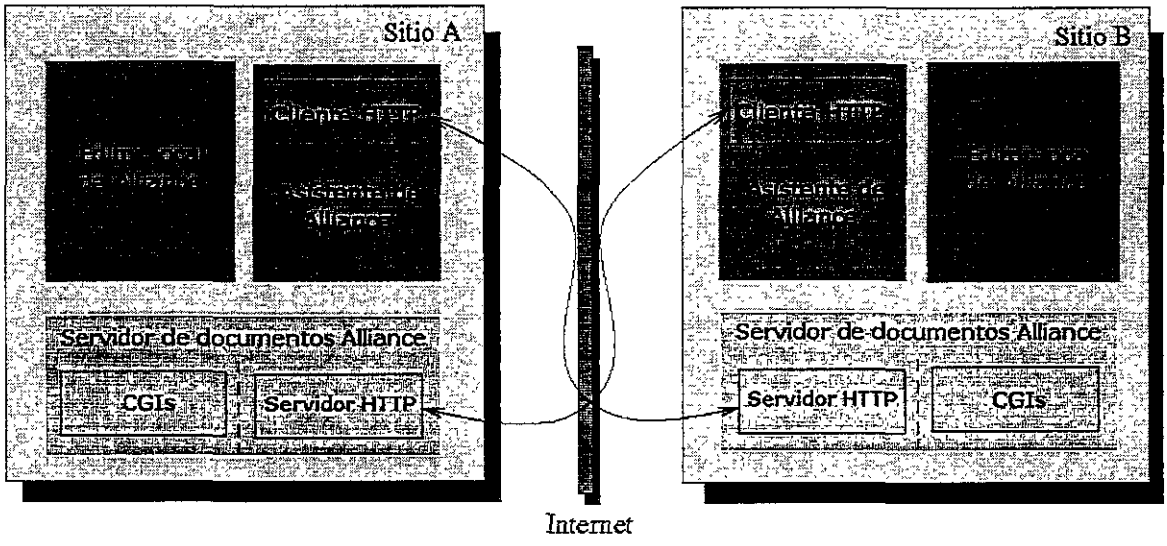


Figura 3.12. Arquitectura de Alliance para la Web.
Fuente: Traducido de [Rom98].

En la figura 3.12 se presenta la arquitectura de Alliance para la Web. Cada sitio en Alliance actúa como cliente de edición (apoyado por una librería HTTP) y como servidor de documentos, apoyado por el servidor HTTP y un conjunto de CGIs desarrollados en el lenguaje C. Los CGIs realizan las operaciones que atienden los requerimientos de los clientes. Dichas operaciones comprenden:

- obtener información sobre los documentos compartidos (lista de autores y grupos, lista de documentos, roles de edición actuales, etc.),
- evaluar un rol de edición para un autor,
- obtener copias locales de las nuevas versiones de los fragmentos,
- llevar a cabo la migración de las copias,
- obtener copias de documentos remotos,
- envío de eventos especiales para intercambiar copias de los fragmentos, bloquear o desbloquear fragmentos, etc.

A diferencia del manejo centralizado de documentos de la arquitectura Web, la arquitectura de *Alliance utiliza el manejo de documentos distribuidos. Este manejo se basa en la copia total de* los fragmentos en cada sitio de cada autor que participa en la edición.

Para asegurar la consistencia de un documento, los fragmentos distribuidos son organizados como copias maestras y esclavas. De esta manera, para cada fragmento existe:

- una sola copia maestra, la cual autoriza la modificación del fragmento,

- diversas copias esclavas que autorizan solamente la lectura del fragmento.

Aún cuando varios autores tengan roles potenciales de escritor o de administrador, solamente el autor que posea la copia maestra del fragmento es quien puede escribir sobre él en forma efectiva. En este caso, los demás autores poseerán copias esclavas de dicho fragmento.

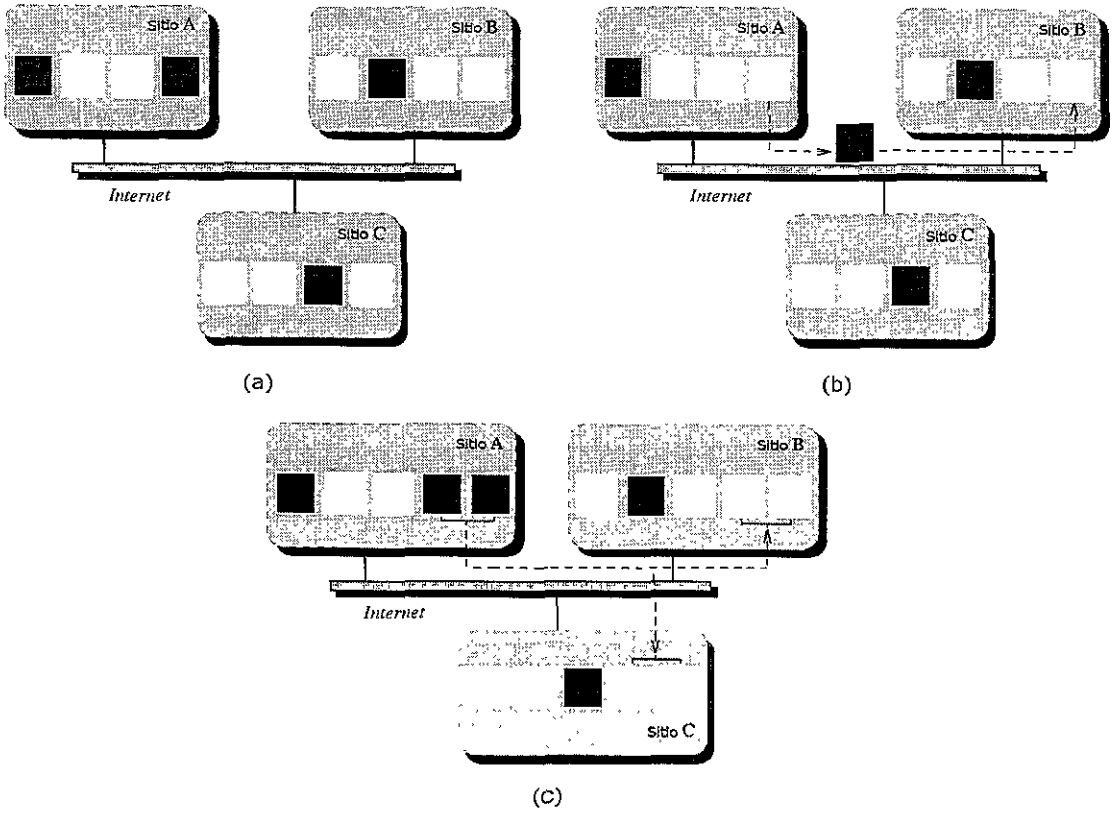


Figura 3.13. Distribución de copias de fragmentos.
(a) Composición de un documento de 4 fragmentos con copias maestras y esclavas.
(b) Migración de la copia maestra del cuarto fragmento.
(c) Actualización de la estructura de un fragmento.
Fuente: Traducido de [Rom98].

En la figura 3.13 se ejemplifica la distribución de las copias de los fragmentos. Las copias maestras se presentan como cuadro resaltados mientas que las copias esclavas se presentan como cuadros blancos. En la parte (a) de la figura se muestra como la versión más actualizada del documento está conformada por los fragmentos primero y cuarto, ubicados en el sitio A, el segundo fragmento ubicado en el sitio B y el tercer fragmento ubicado en el sitio C. De esta manera, la versión más actualizada del documento es vista como el conjunto de todas las copias maestras distribuidas.

Para que los autores puedan contribuir en diferentes fragmentos las copias maestras deben migrar de un sitio a otro. Esta migración se lleva a cabo cada vez que el autor decide transferir su rol de escritor o de administrador a otro autor. En la figura 3.13 (b) se muestra gráficamente la migración de la copia maestra del cuarto fragmento entre los autores A y B.

Cuando un autor posee la copia maestra de un fragmento no sólo puede modificar su contenido sino también su estructura. En la figura 3.13 (c) se muestra la actualización de la

estructura del cuarto fragmento y su distribución en el momento en que el autor del Sitio A decide validar sus cambios. Nótese que las copias de los nuevos fragmentos, generados a partir del cuarto fragmento, se distribuyen como copias esclavas, mientras que las copias maestras permanecen en el sitio donde se realiza el cambio de la estructura, hasta que se decida transferir el rol de escritor o administrador a otro autor.

Para obtener una copia remota de un documento, el asistente de Alliance genera un requerimiento *HTTP-POST* el cual contiene el URL del servidor de documentos remoto, el nombre de acceso del usuario y el nombre del documento. El servidor de documentos genera un evento de respuesta a este requerimiento y ejecuta el respectivo CGI asociado a él. Dicho CGI transforma la estructura de árbol del documento, en una cadena que se envía utilizando la respuesta del *HTTP-POST*. Finalmente, esta cadena es recibida por el asistente de Alliance, el cual reconstruye la estructura de árbol y transfiere el control al editor local para su visualización.

Debido a que Internet es un medio sujeto a diversas fallas, Alliance implementa un administrador de transacciones que garantiza la confiabilidad en el proceso de migración de las copias maestras. Este administrador de transacciones implementa mecanismos para el control de la consistencia, tales como deshacer (*"Rollback"*), recuperar, verificación y certificación, y mantiene registros duplicados del estado de cada transacción realizada para cada fragmento. El administrador de transacciones se ejecuta solamente en el sitio que requiere la copia maestra del fragmento.

Para realizar el proceso de migración se consideran los siguientes aspectos:

- Cada sitio posee un identificador único denominado **siteAddr**.
- Cada fragmento de cada sitio (sea copia maestra o esclava) posee un identificador único denominado **FragId**.
- Cada fragmento que corresponde a una copia esclava posee un número de versión.

La migración de la copia maestra consiste en cambiar la naturaleza del fragmento (maestro o esclavo), tanto en el sitio que requiere la copia como en el sitio que la libera. Adicionalmente, el proceso de migración verifica la versión de la copia esclava, para determinar si es necesario actualizar su contenido previamente al cambio de la naturaleza del fragmento.

Para cambiar la naturaleza del fragmento se define la función SetFragNat(FragId,TypeNature), la cual asigna la naturaleza TypeNature (maestro o esclavo) al fragmento identificado por FragId. Esta función incluye una restricción de integridad mediante la función GetFragNat(siteAddr, FragId)=0, si el fragmento FragId del sitio siteAddr es una copia esclava y GetFragNat(siteAddr, FragId)=1, si el fragmento FragId del sitio siteAddr es una copia maestra. De esta forma, la restricción impuesta para la naturaleza del fragmento se define como:

$$\mathbf{XOR(GetFragNat(siteAddr_1, FragId), GetFragNat(siteAddr_2, FragId))=1}$$

Con ello, el mecanismo de verificación del administrador de transacciones garantiza que las copias maestras y esclavas pasen de un estado inicial consistente a un estado final consistente y que no existan copias maestras duplicadas.

Durante la migración de la copia maestra el administrador de transacciones realiza una serie de estados, los cuales se escriben en los registros de estado de la transacción, junto con la operación que debe efectuarse en caso de falla. Estos estados y las operaciones se definen como: Inicio (operación deshacer), Ejecución (operación recuperar), Terminación (operación

verificación) y Compromiso (operación verificación). Cada vez que se recupera de una falla, el administrador de transacciones verifica su registro de estado y continúa en el último estado registrado, hasta alcanzar el estado de compromiso. Con ello, el proceso de migración es confiable y no se presentan inconsistencias en las copias distribuidas de los fragmentos.

3.4. Otros sistemas de edición colaborativa en Internet

En los desarrollos de sistemas colaborativos, se ha adoptado con mayor frecuencia una arquitectura híbrida, combinando procesos centralizados y distribuidos para el almacenamiento de documentos y el manejo de la consistencia. A continuación se revisan los editores colaborativos en Internet más representativos y sus características.

3.4.1. SEPIA (*"Structured Elicitation and Processing of Ideas for Authoring"*)

SEPIA[4] [HaWi92] es un sistema para la de edición colaborativa de documentos hipertexto. El proceso de edición se provee mediante cuatro espacios de actividades: planeación, contenido, discusión y retórica. Las tareas que se llevan a cabo sobre estos espacios comprenden: planeación, estructuración, discusión y escritura.

SEPIA permite a los autores trabajar sobre diferentes partes de un documento en forma asíncrona o sobre la misma parte en forma síncrona. Para ello, los autores pueden trabajar en tres modos diferentes: individual, débilmente acoplado y fuertemente acoplado.

En los espacios de actividad se proveen telepunteros y barras de estado para indicar las operaciones que se están realizando en modo fuertemente acoplado. Las barras de estado proveen acceso a sistemas de conferencia de audio en tiempo real para el intercambio de mensajes.

Por otra parte, en SEPIA se incorporan marcas de diferentes colores sobre los objetos hipermedia para indicar las actividades realizadas en modo débilmente acoplado. Los modos de edición individual y colaborativa proveen la misma interface (espacios de actividad) y una operación de sincronización es llevada a cabo para facilitar la transferencia entre ellos.

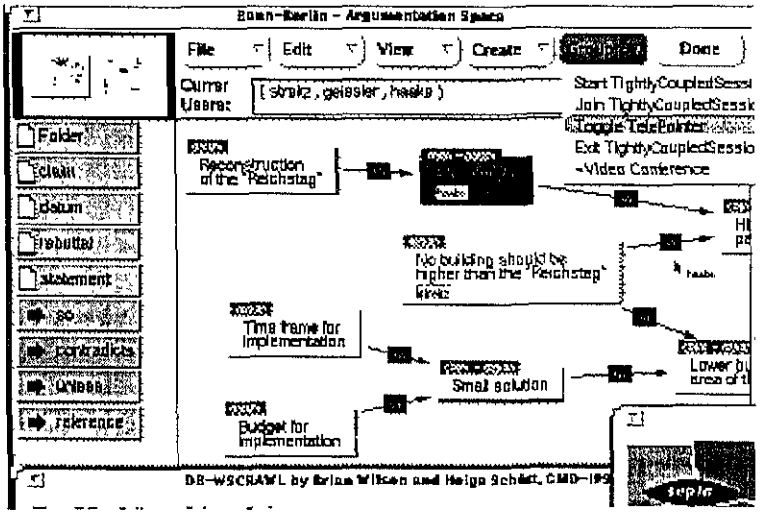


Figura 3.14. Interface del sistema SEPIA.
Fuente: www.darmstadt.gmd.de/concert/activities/past/sepia.html.

[4] SEPIA http://www.darmstadt.gmd.de/concert/activities/past/sepia.html

En la figura 3.14 se presenta un ejemplo de la interface de SEPIA. En este ejemplo se muestra la estructura de un documento por medio de ligas de sus divisiones. En la esquina superior izquierda se maneja una pequeña ventana que indica la parte del documento que está siendo visualizada (vista radar). Además, en el ejemplo se encuentra seleccionada la opción **Grupo** del menú principal (cuadro superior a la derecha), desde la cual se invocan las herramientas de colaboración, en los modos débil y fuertemente acoplado.

SEPIA está desarrollado para redes de estaciones de trabajo Unix y utiliza una arquitectura híbrida, donde los documentos son copiados en los diversos clientes. Su almacenamiento se basa en el motor de documentos hipermedia HyperBase [ScSt90] sobre Sybase[5] y se utilizan mecanismos de bloqueo de objetos en la base de datos para el control de la concurrencia y la sincronización de las copias. En los sitios servidores de bases de datos se incorpora un servidor de difusión de eventos, el cual se encarga de la notificación a todos los clientes y coordina el proceso de sincronización y bloqueo de documentos.

3.4.2. SASSE (*"Synchronous Asynchronous Structured Shared Editor"*)

SASSE [Bae93][Bae94][Nas92] es un sistema de edición colaborativa basado en la organización estructurada de las actividades de edición. Este editor provee un espacio de trabajo mediante un documento compartido. Los autores colaboran sobre una vista WYSIWIS sincronizada del documento la cual es común para todos los colaboradores. SASSE provee los modos de interacción síncrono (fuerte y débilmente acoplado) y asíncrono.

En este editor se proporcionan telepunteros y secciones remarcadas de diversos colores, para representar el trabajo que cada autor hace sobre el documento y evitar conflictos durante la interacción fuertemente acoplada. Asimismo, este editor provee una barra de estado que indica la ubicación de cada autor que trabaja en modo débilmente acoplado. Para facilitar la conciencia de grupo en modo síncrono SASSE permite la comunicación de los autores mediante herramientas de audio y video.

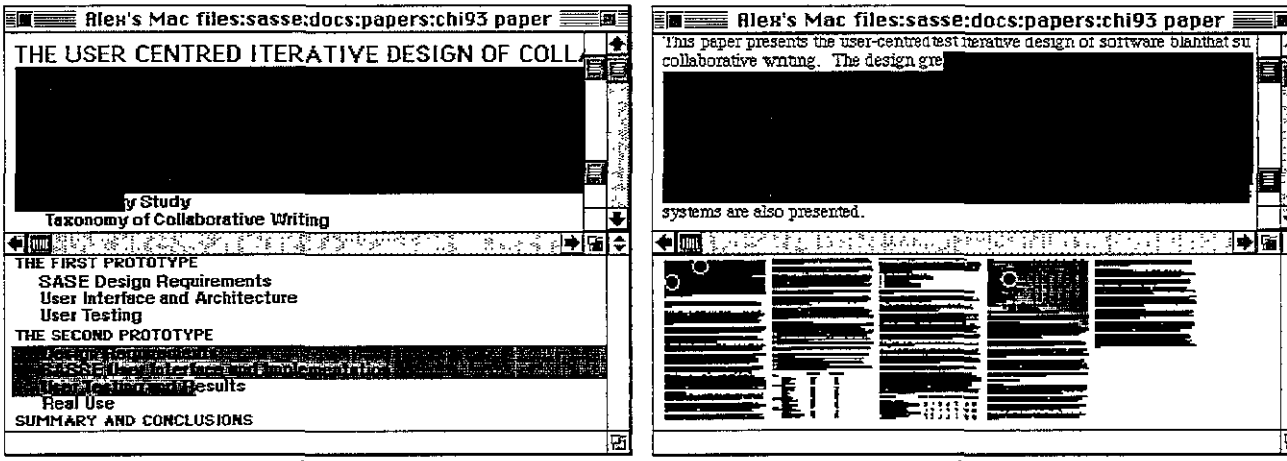


Figura 3.15. Interfaces de SASSE. (a) Vista de la sección editada del documento y la actividad de otro usuario en particular. (b) Vista del documento editado y las ubicaciones de los demás autores en el documento. Fuente: [Bae93].

[5] Motor de base de datos relacional, http://www.sybase.com/.

En la figura 3.15 se presenta un ejemplo de la interface de SASSE. La parte (a) muestra el caso de la vista del documento editado (división superior) y la actividad de otro usuario sobre el documento (división inferior).

La parte (b) de esta figura, presenta la sección editada del documento (división superior) y la vista general de documento, con las ubicaciones de los demás autores sobre las secciones correspondientes. Nótese el uso de diferentes colores para la identificación de los usuarios y el entendimiento de sus actividades.

En SASSE se provee la interacción en modo asíncrono mediante un mecanismo de verificación de versiones del documento. Además, se usan diferentes colores para resaltar el nuevo texto que ha sido agregado o que está listo para su incorporación. Estos mecanismos son controlados por los autores en diferentes instantes de tiempo.

SASSE utiliza una arquitectura distribuida, con un mecanismo de almacenamiento distribuido de documentos, sobre los cuales se realizan las operaciones de verificación de versiones, escritura, borrado, etc. La vista del documento es actualizada en el momento en que el autor decide sincronizarla. A partir de ello, los eventos que ocurren son difundidos en tiempo real entre todas las vistas de los autores.

El sistema SASSE está desarrollado para una red de estaciones de trabajo Macintosh la cual puede ser local o de gran distancia. Por otro lado, este sistema ha servido de base para un nuevo prototipo experimental llamado Calliope[6] [Mit96], que se está desarrollando actualmente en la Universidad de Toronto. Este prototipo facilita la edición colaborativa síncrona como medio para la comunicación y el aprendizaje compartido del proceso de edición.

3.4.3. Prep (*"work in Preparation editor"*)

Prep [Neu90][Neu94] es un editor colaborativo asíncrono que facilita la adición de comentarios a un documento en forma concurrente. Este editor fue desarrollado para una red de estaciones de trabajo Macintosh y diseñado para apoyar tres aspectos fundamentales: la interacción social (establecimiento de roles para la coordinación de las actividades), las diferentes fases del proceso de edición y los tipos de interacción (síncrona y asíncrona) entre los autores de un documento.

Un documento es definido en Prep como un conjunto de una o más columnas, las que a su vez se definen como un conjunto de cadenas. Esta estructuración permite a los autores construir una red de conceptos a partir de la agrupación de columnas, mediante ligas y cadenas.

Las primeras dos columnas se utilizan para la definición y la planeación del documento (texto principal del documento) y para el contenido del mismo. En las columnas siguientes se presentan en forma síncrona, por cada autor, los comentarios y las anotaciones que ellos realizan.

Las cadenas en la columna del texto principal corresponden a los párrafos del texto y en las demás columnas, corresponden a cada comentario de cada autor. Además, las cadenas pueden referirse también a estructuras de árbol de otras cadenas o a imágenes.

[6] Editor compartido multiusuario. http://www.dgp.toronto.edu/people/alex/thesis/calliope.html

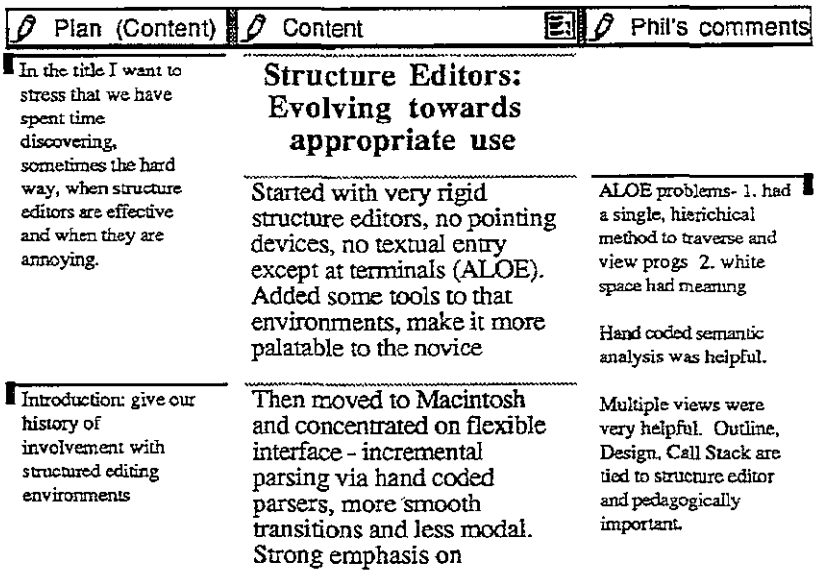


Figura 3.16. Documento en Prep.
Fuente: [Neu90].

En la figura 3.16 se presenta un ejemplo de un documento en el editor Prep. La primer columna presenta el plan general del documento. La segunda columna presenta su contenido y la tercera columna presenta los comentarios realizados por un autor específico. Nótense las divisiones entre los párrafos de la segunda columna, para indicar la sección del texto al que hacen referencia los comentarios de la tercera columna.

Prep utiliza una arquitectura híbrida, donde cada autor mantiene una copia del documento original y las anotaciones se realizan directamente sobre cada copia. Un autor puede identificar los cambios entre su copia y las versiones del documento original, para decidir si los incorpora en su versión actual.

La comunicación entre los autores se lleva a cabo a través del plan del documento y de los comentarios que se agregan durante el proceso de edición. De esta manera, durante la planeación del documento los autores comparten sus ideas y generan nuevas cadenas (párrafos) relacionadas con la estructuración del documento.

El control de los documentos se lleva a cabo en una base de datos centralizada, donde además se establecen filtros para la comunicación de eventos. Con base en los roles establecidos (lector, revisor, escritor), Prep controla las acciones que los autores pueden realizar. De esta forma, las peticiones de los autores son analizadas por los filtros de la base de datos para determinar la información que se accesa y que debe transmitirse.

3.4.4. Quilt (*"rich set of connotations"*)

Quilt [Fis88] es un sistema de edición colaborativa asíncrono que combina hipermedia multiusuario, un sistema de conferencia y correo electrónico multimedia. Este sistema se desarrolló para una red de estaciones de trabajo Unix y permite que los usuarios compartan documentos, mensajes de voz y de texto.

La base para la colaboración son las anotaciones que hace cada autor sobre las diferentes secciones de los documentos. Para ello, en Quilt se incorpora el establecimiento de roles, los cuales determinan los privilegios (derechos) que tiene cada autor sobre los documentos en

cada instante del proceso de edición. Además, los roles se usan para construir las vistas de los documentos con la información disponible para los autores.

Los documentos se estructuran utilizando hipermedia, mediante la cual un objeto de información (un párrafo, una imagen, un archivo de audio) es definido como un nodo y se crean ligas entre los nodos para conformar el documento completo. Un documento en Quilt consiste en tres secciones:

- El documento base actual (texto, gráficos, tablas) que cada autor comparte públicamente con los demás coautores.
- Las sugerencias de revisión con las cuales los autores, con los privilegios adecuados, pueden agregar objetos de información al documento base.
- Comentarios de voz o texto.

La coordinación entre los autores se provee mediante dos mecanismos: un registro de las actividades que realiza cada autor sobre cada nodo y el intercambio de mensajes estructurados (texto y voz) como un sistema de conferencia. Además, el registro de actividades es usado para el control de la concurrencia sobre los documentos.

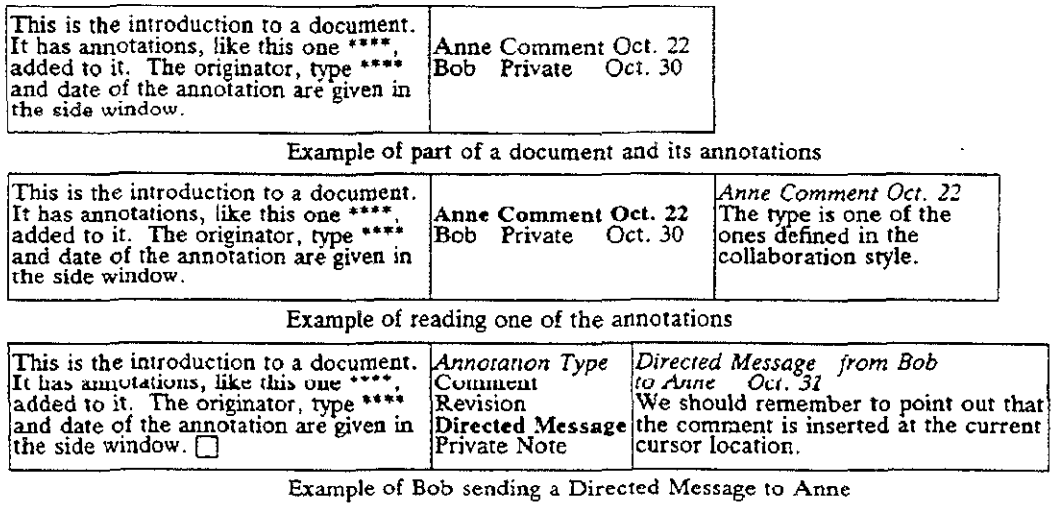


Figura 3.17. Ejemplo de un documento y mensajes estructurados en Quilt. Fuente: [Fis88].

La figura 3.17 presenta un ejemplo de un documento estructurado en Quilt. El recuadro superior muestra el contenido del documento y las ligas a las anotaciones realizadas por dos autores. El recuadro central muestra la lectura de una de las anotaciones en la parte derecha del recuadro. El recuadro inferior muestra la adición de una anotación de mensaje de texto del usuario **Bob** dirigido al usuario **Anne**.

Quilt utiliza una arquitectura centralizada para el almacenamiento y la colaboración a través de un servidor de documentos. En este servidor se almacena la información sobre los documentos, los privilegios de los autores y la colaboración llevada a cabo mediante las anotaciones. Los eventos son difundidos a los colaboradores en forma centralizada, mediante accesos a los privilegios y a las ligas de los nodos almacenados en el servidor.

3.4.5. GROVE (*"GRoup Outline Viewing Editor"*)

GROVE [ElGi89] es un editor colaborativo síncrono que permite la edición de documentos estructurados mediante el uso de vistas grupales de tipo WYSIWIS, que son distribuidas en los sitios clientes.

Estas vistas de grupo pueden ser privadas, compartidas y públicas. El documento es presentado en cada vista mediante una estructura de árbol, donde los niveles de mayor jerarquía representan las divisiones del documento.

Los autores interactúan directamente sobre la estructura del documento, realizando las operaciones de insertar, borrar, cortar y pegar texto, etc. Además, los autores utilizan la estructura del documento para asignar privilegios de lectura y escritura sobre las divisiones.

En GROVE se emplean múltiples vistas para la representación de las divisiones de cada documento. De igual forma, en la vista de grupo se representa el estado de la interacción de cada autor con cada división. Esta forma de conciencia de grupo es controlada con mecanismos para el bloqueo y el establecimiento de un orden secuencial de las operaciones (serialización).

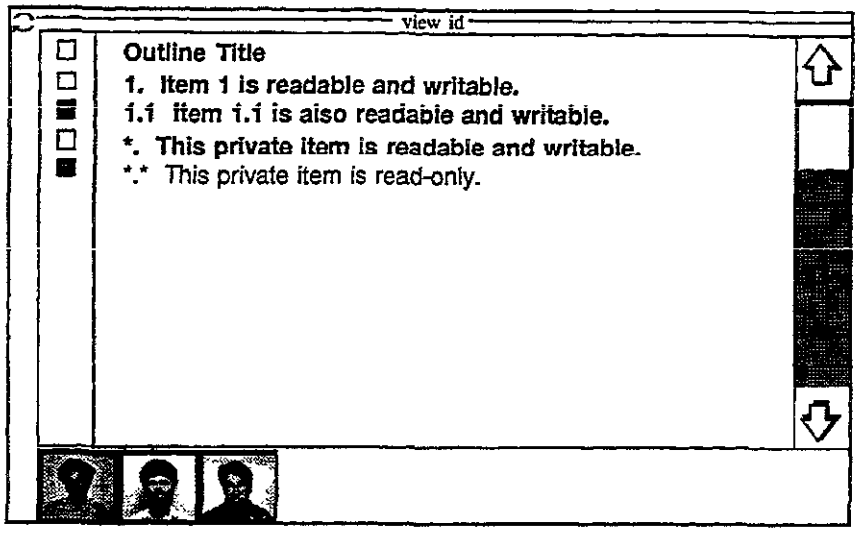


Figura 3.18. Ejemplo de una ventana de grupo en GROVE.
Fuente: [ElGi89].

En la figura 3.18 se ejemplifica la vista de grupo de GROVE. En esta vista, tres autores están estableciendo la estructura del documento en forma conjunta, al igual que la asignación de los privilegios de lectura y escritura sobre el elemento **1.1** en particular.

El sistema GROVE utiliza una arquitectura híbrida donde cada vista es copiada entre todos los autores activos. En cada cliente existe un proceso para la comunicación con un servidor centralizado que coordina la serialización de las operaciones generadas en cada sitio. Una vez que el servidor establece el orden en que deben llevarse a cabo las operaciones, éstas son distribuidas en los clientes para actualizar sus vistas.

GROVE se desarrolló sobre una red de estaciones de trabajo Unix y utiliza un sistema externo de conferencia de audio para la comunicación directa entre los usuarios.

3.4.6. COARSY (*"COllaborative Asynchronous Review System"*)

COARSY [RuFa98][Ruiz98] es un sistema colaborativo asíncrono para la edición de documentos HTML. Este sistema se basa en el modelo de discusión o modelo conversacional [ShBa93], en el cual los componentes de una discusión (preguntas, respuestas, anotaciones, comentarios) y sus relaciones, permiten la estructuración y la revisión de un documento en forma colaborativa.

En COARSY, los autores visualizan la estructura del documento, el contenido de cada división y la discusión relacionada sobre este contenido. Para ello, la interface del sistema presenta dos secciones: para el documento estructurado y para la discusión estructurada.

Los autores realizan sus contribuciones únicamente en la sección de discusión, debido a que el sistema no permite escrituras concurrentes. Solamente el autor dueño del documento es quien realiza las respectivas modificaciones, apoyado en las discusiones generadas con los demás coautores.

De esta forma, COARSY establece los roles de autor (escritor) y revisor (contribuye a la discusión), para facilitar su interacción y determinar los accesos a las discusiones que se llevan a cabo durante la producción del documento.

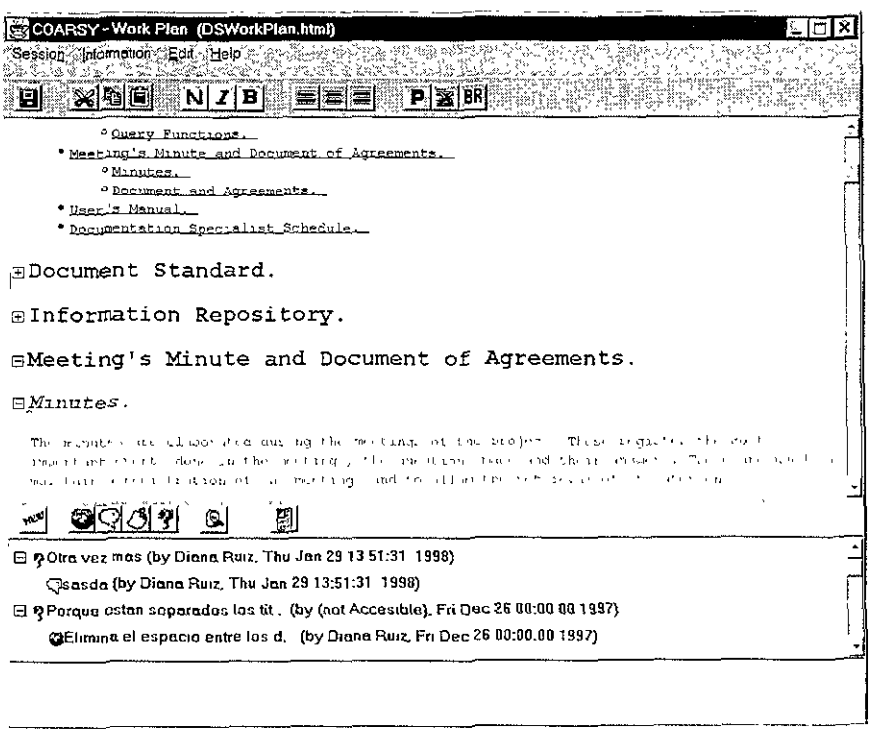


Figura 3.19. Interface principal del sistema COARSY.
Fuente: [RuFa98].

La figura 3.19 presenta la interface del sistema COARSY, la cual es visualizada por todos los autores que participan en la discusión. La división superior de la ventana presenta el contenido del documento y su estructura. La división inferior presenta la sección de discusión sobre los componentes del documento. En el ejemplo se muestran los hilos de discusión **Otra vez más** y **Porque están separados los tit....**, los cuales tiene a su vez asociados mensajes y comentarios.

COARSY está basado en una arquitectura híbrida de tres capas, utilizando un servidor para el sistema y otro para la base de datos donde se almacenan los documentos (servidor centralizado). Los autores accesan los documentos mediante peticiones al servidor del sistema y éste dirige las peticiones al servidor de base de datos para que el documento sea copiado hacia cada autor.

El sistema COARSY se encuentra en una etapa experimental y tiene la ventaja de ser desarrollado en Java, lo cual lo hace portable sobre diferentes plataformas y sistemas operativos (Unix Solaris, Windows, Linux).

3.4.7. Comparación de Alliance con otros editores colaborativos en Internet

Los criterios utilizados para la comparación del editor colaborativo Alliance con otros editores colaborativos similares, comprenden las características de las fases del proceso de edición que son facilitadas, los modos de interacción, el manejo de roles para el grupo de autores, los mecanismos de conciencia de grupo y la arquitectura con la cual se apoya la producción de documentos en forma conjunta.

Las siguientes tablas sintetizan las características de los editores colaborativos tratados a lo largo del presente capítulo.

Sistema	Fases del proceso de edición	Modos de Interacción	Roles
ALLIANCE	- Planeación y consolidación individual. - Escritura colaborativa.	- *Modo autónomo.* - Escritura asíncrona. - Visualización en forma síncrona débilmente acoplada	Escritor, lector, nulo y administrador
SEPIA	Planeación y escritura colaborativa.	- Escritura asíncrona. - Discusión, visualización y comunicación en forma síncrona, fuerte y débilmente acoplada.	Autor
SASSE	Planeación, escritura y consolidación colaborativa.	- Escritura asíncrona y síncrona débilmente acoplada. - Comunicación síncrona fuertemente acoplada - Visualización síncrona débilmente acoplada	Autor.
PREP	Planeación, escritura y consolidación colaborativa.	- Escritura asíncrona. - Anotaciones concurrentes en forma síncrona débilmente acoplada.	Escritor, lector y revisor
QUILT	- Planeación colaborativa. - Escritura individual.	- Escritura asíncrona y anotaciones síncronas. - Visualización en forma síncrona débilmente acoplada.	Autor, lector y revisor
GROVE	Planeación y escritura colaborativa.	- Escritura asíncrona y síncrona débilmente acoplada. - Visualización síncrona fuerte y débilmente acoplada.	- Lector y escritor - Privilegios de modificación.
COARSY	- Planeación colaborativa. - Escritura individual.	Discusiones y anotaciones en forma asíncrona	Autor y revisor.

Tabla 3.1. Características de las fases del proceso de edición, los modos de interacción y el manejo de roles de los editores colaborativos en Internet.

Como se muestra en la tabla 3.1, los editores colaborativos están orientados a facilitar principalmente las fases de planeación y escritura de los documentos. En el caso específico de Alliance, la planeación se lleva a cabo de manera individual, debido a que solamente el dueño

del documento (rol de administrador) es quien establece la estructura y la asignación de roles, desde el instante de la creación del mismo.

Esto constituye una desventaja de Alliance respecto a los demás editores de la tabla 3.1, ya que como se ha visto, la fase de planeación debe involucrar las opiniones y las contribuciones del grupo de autores desde el inicio del proceso de edición.

En el caso de los editores Quilt y COARSY, la edición del documento se lleva a cabo en forma individual porque los autores no contribuyen directamente sobre su contenido. Sin embargo, estos editores proporcionan otras alternativas para la colaboración, como lo son el uso de anotaciones y discusiones síncronas o asíncronas, para que los autores interactúen en la revisión del documento.

En Alliance, la interacción síncrona débilmente acoplada se lleva a cabo mediante la visualización de las nuevas versiones de los fragmentos del documento y el cambio de estado de los iconos de rol. Esta alternativa de colaboración no es suficiente para facilitar la negociación espontánea entre los autores puesto que no establece una forma de comunicación entre ellos.

El manejo de diversos roles para los autores beneficia la coordinación del trabajo y la seguridad de la información de los documentos compartidos. En Alliance, el manejo de roles se utiliza también para simplificar el control de las escrituras concurrentes y el proceso de migración de las copias maestras de los fragmentos.

Sistema	Mecanismos de conciencia de grupo	Arquitectura
ALLIANCE	- Vistas WYSIWYG con iconos representativos de roles y acciones permisibles - Cambios de estado de los iconos	- Fragmentación de documentos. - Arquitectura distribuida para los servidores de documentos y el control de las réplicas.
SEPIA	- Telepunteros y barras de estado. - Vistas WYSIWIS y WYSIWYG con vistas radar. - Sesiones síncronas fuerte y débilmente acopladas	- Híbrida, con manejo de documentos distribuidos. - Control de concurrencia centralizado.
SASSE	- Telepunteros y barras de usuarios - Secciones de texto marcadas con diferentes colores para cada usuario - Vista WYSIWIS	- Distribuida, con manejo de versiones de documentos - Difusión de eventos en forma síncrona.
PREP	Interface WYSIWYG con columnas de anotaciones.	- Híbrida con réplicas de documentos y anotaciones. - Control centralizado para el acceso y la difusión de eventos.
QUILT	- Anotaciones. - Comunicación asíncrona mediante e-mail y conferencia de audio - Registro de actividades.	Almacenamiento y difusión de eventos en forma centralizada.
GROVE	Vistas WYSIWIS grupales, públicas y privadas.	- Híbrida, con réplicas de documentos y de vistas. - Coordinación centralizada con mecanismos de serialización de operaciones.
COARSY	- Iconos de discusión en forma asíncrona - Registro de las contribuciones a las discusiones	Arquitectura de la Web extendida con el modelo de tres capas

Tabla 3.2. Mecanismos de conciencia de grupo y arquitecturas de los editores colaborativos en Internet.

De la tabla 3.2 se destaca que Alliance promueve la conciencia de grupo mediante los estados de los iconos asociados a los fragmentos. Este mecanismo facilita que un autor se entere de una nueva versión de un fragmento y decida si acepta o no visualizarla en su interface del documento. Sin embargo, este mecanismo no permite que el autor, quien se entera de la nueva versión del fragmento, pueda discutir con el autor quien realiza esa nueva versión.

Asimismo, la interface de Alliance no incorpora información sobre la ubicación de los autores, ni de cual autor escribe o trabaja sobre cual fragmento, sino que es necesario que cada autor defina esta relación desde la creación del documento.

En los editores SEPIA, SASSE y GROVE, las vistas WYSIWIS, los telepunteros, las barras de usuarios y las vistas radar, proporcionan mayor información a los autores sobre su ubicación y el estado de sus actividades. Esta información es relevante puesto que facilita una interacción espontánea sobre determinada sección o estructura del documento.

Por otra parte, el manejo de copias de los documentos constituye la alternativa principal para facilitar la edición colaborativa en Internet. Como en el caso de Alliance, el manejo distribuido de dichas copias no sólo permite la escritura en forma individual y la consistencia de los documentos, sino que agrega un mayor grado de tolerancia a fallas en situaciones donde los autores se encuentran ubicados a gran distancia.

3.5. Discusión

En el presente capítulo se revisaron las características de la edición de documentos como actividad colaborativa y los requerimientos que surgen a partir de ellas, para los editores colaborativos que utilizan Internet como medio de comunicación.

De igual forma, se han revisado las principales características del editor Alliance, respecto a la estructuración de documentos en fragmentos, el apoyo a la conciencia de grupo mediante iconos con diferentes estados y la arquitectura extendida de la Web, utilizada para la distribución de las copias de los documentos.

En el desarrollo de sistemas de edición colaborativa se han explorado diversas alternativas para representar un documento como el medio para la colaboración. Asimismo, en estos editores se busca facilitar la interacción síncrona de los autores, mediante herramientas externas de comunicación o mediante las funciones del editor mismo (anotaciones concurrentes, discusiones, etc.). Esto resalta el hecho de que a pesar de que la mayor parte del trabajo de edición se realiza en forma asíncrona, los editores colaborativos deben ser flexibles y facilitar la transición a otras formas de interacción entre los autores.

En relación con los desarrollos de los editores colaborativos, se ha utilizado con mayor frecuencia una arquitectura híbrida con distribución de copias de los documentos. Esta arquitectura no sólo facilita el proceso de escritura individual, sino que adiciona características para la tolerancia a fallas en sistemas a gran escala. A pesar de ello, el problema del manejo distribuido de las copias de los documentos continúa siendo tema de investigación, en el cual se buscan nuevas arquitecturas que utilicen en forma más eficiente la comunicación a través de Internet y la Web.

En los editores colaborativos revisados en este capítulo, las interfaces de usuario incorporan herramientas de conciencia de grupo como vistas WYSIWIS y WYSIWYG, telepunteros, registros de actividades, etc., para que los autores coordinen sus actividades. De igual forma, estos editores utilizan formatos específicos para la estructuración y el contenido de los documentos.

La evolución de la arquitectura Web ha hecho posible la utilización de los navegadores estándar como interface de usuario y la manipulación directa de documentos estructurados en los lenguajes HTML y XML. Editores como Amaya[7], Coffee Cup[8] y Vervet Logic[9], utilizan vistas WYSIWYG para la edición de estos documentos y facilitan su publicación en línea en los servidores Web. Sin embargo, en este tipo de editores aún no se incorporan herramientas para la interacción entre grupos de autores que trabajan en forma colaborativa.

Para el caso del editor Alliance, en la actualidad existen dos proyectos de investigación que pretenden extender sus características para ajustarlo a las tendencias de la edición colaborativa en la Web:

- **AllianceWeb** [Dec99]. En este proyecto se busca integrar una arquitectura de 3 capas denominada PIÑAS [Dec01], para la edición colaborativa de documentos PIV (de Alliance) y HTML, directamente desde un navegador. Además, en esta arquitectura se combina el modelo de discusiones de COARSY y un esquema URL extendido para la designación de los documentos y otros objetos como imágenes, anotaciones, etc.

- **BYZANCE**[10]. Es un proyecto del Instituto Francés de Investigaciones en Ciencias de la Computación y Control (INRIA[11]). En este proyecto se busca combinar las facilidades del editor Amaya para la estructuración de documentos HTML, con el manejo de documentos distribuidos de Alliance.

Las características y limitaciones del editor Alliance sugieren el desarrollo de herramientas para proveer la comunicación directa y facilitar la negociación en forma síncrona, entre los autores. Con estas herramientas, las actividades individuales de edición llegarían a ser más flexibles y los autores de un documento podrían contribuir con sus opiniones durante todas las fases de dicho proceso colaborativo.

El presente trabajo de investigación propone la implementación de estas herramientas dentro de un espacio de trabajo compartido, que beneficie la interacción y promueva la conciencia de grupo en las actividades de edición. Este espacio compartido debe apoyarse en una arquitectura abierta que pueda integrarse fácilmente a la infraestructura de la Web.

[7] http://www.w3.org/Amaya/.
[8] Editor HTML Coffee Cup, http://www.coffeecup.com/editor/.
[9] Vervet Logic XML Pro, http://www.vervet.com/xmlpro.html.
[10] Proyecto Opera-Byzance, http://hatutu.inrialpes.fr/OPERA/byzance.html
[11] http://www.inria.fr/

Capítulo 4

Desarrollo del espacio de trabajo compartido

En los anteriores capítulos se analizaron las características de los espacios de trabajo compartidos, con las cuales se busca facilitar la interacción entre las personas para realizar una tarea en conjunto. De igual forma, se analizaron las características de la edición colaborativa y se determinó la necesidad de implementar mecanismos de colaboración que mejoren la producción de documentos en el editor colaborativo Alliance.

El presente capítulo aborda el desarrollo del espacio de trabajo compartido en Internet, para el editor Alliance. Este espacio de trabajo está basado en el concepto de sesión como el punto de reunión de los diferentes autores. Asimismo, el espacio de trabajo proporciona tres herramientas colaborativas, que apoyan la fase de negociación del proceso de edición de documentos y facilitan la comunicación entre los autores en forma síncrona. Dichas herramientas comprenden: las discusiones sobre diversos temas, la votación sobre ideas en particular y el intercambio de mensajes.

El desarrollo del espacio de trabajo compartido comprende las etapas de análisis, diseño y construcción, en las cuales se aplica la metodología unificada para el desarrollo de software orientado a objetos, utilizando el lenguaje UML ("*Unified Modeling Language*") [Boo99]. Este lenguaje define una serie de productos (diagramas, modelos, especificaciones, etc.) para cada etapa, los cuales no sólo facilitan la definición de los requerimientos para proveer la interacción entre los autores, sino la identificación y el comportamiento de las clases (elementos básicos) que conforman el espacio de trabajo compartido.

Adicionalmente, el uso de UML, presenta tres beneficios principales para la presente investigación:

- Es un lenguaje flexible, en el cual se utilizan sólo los productos que se necesitan. Por ello, para el desarrollo del espacio de trabajo se utilizan: el **diagrama de casos de uso,** para la definición de los requerimientos, los diagramas de **clases y responsabilidades**, para el análisis de los componentes, los **diagramas de secuencia**, para establecer el comportamiento de las clases, y la **especificación de las clases** para su construcción.
- El análisis de un sistema colaborativo es similar al de un sistema convencional (un solo usuario). Para ello, la definición de clases y los diagramas que establecen su comportamiento, no sólo permiten caracterizar una instancia simple del sistema, sino también la concurrencia.
- En la programación orientada a objetos se facilita el uso de librerías de propósito específico, como las proporcionadas por el lenguaje de programación Java, para las interfaces gráficas del espacio de trabajo, y el API Java Shared Data Toolkit (JSDT), para la comunicación de grupos.

El desarrollo del espacio de trabajo compartido para Alliance se basa en una arquitectura híbrida, que adopta los componentes del JSDT para el manejo centralizado de la sesión de grupo y la difusión de eventos y mensajes en forma distribuida. Esta arquitectura es

independiente de los protocolos de comunicación (TCP/IP, HTTP, etc.) y permite la adición de nuevos componentes e interfaces gráficas, para extender los mecanismos de colaboración del espacio de trabajo.

Finalmente, en el presente capítulo se realiza la evaluación del espacio de trabajo compartido para Alliance, considerando los siguientes aspectos:

- Las características generales que presentan los espacios de trabajo compartidos.
- El desempeño de la arquitectura en un esquema de conexión en Internet.

4.1. Conceptos básicos

En cualquier metodología orientada objetos se establecen las clases como el elemento básico sobre el cual se realiza el análisis de un sistema. En [Boo99] una **clase** se define como la descripción de un conjunto de objetos que comparten los mismos atributos, sus operaciones y las relaciones con otros objetos. Un **objeto** es cualquier cosa que tenga asociado un conjunto de propiedades (atributos) y un comportamiento (métodos), que lo hacen totalmente distinguible de otros objetos. De esta forma, una clase es una descripción de atributos y métodos, mientras que un objeto es una instancia de una clase.

En UML se introduce otro concepto especial denominado interface (diferente al término interface gráfica de usuario). Una **interface** se define como un conjunto de atributos y operaciones que se utilizan para declarar el comportamiento general de una clase. Sin embargo, a diferencia de una clase, la interface no implementa el comportamiento del método, sino que simplemente lo declara. Por ello, para utilizar una interface, es necesario establecer una clase, en la cual se implementen las operaciones de los métodos declarados en la interface.

Las clases no existen de manera aislada dentro del sistema. Cada clase establece un tipo de relación para llevar a cabo sus funciones y colaborar o interactuar con las demás clases. En UML, se definen principalmente los siguientes tipos de relaciones entre clases:

- **Dependencia**: es una relación que indica que el cambio en la especificación de una clase afecta las clases que la usan. Esta relación es muy común cuando existen clases que utilizan librerías de elementos, para generar una interface gráfica de usuario.

- **Asociación**: es una relación estructural que especifica que un objeto de una clase está conectado con otros objetos de otras clases, para colaborar entre sí. Esta relación posee una propiedad que indica el número de objetos que puede darse en dicha asociación y que se define como **multiplicidad**. La multiplicidad indica que cero, uno o más objetos de una clase, están asociados a cero, uno o más objetos de las demás clases.

- **Generalización**: es una relación entre una clase general, denominada superclase o clase padre, y una clase más específica, denominada subclase o clase hija. En esta relación se indica que la clase hija puede ser usada en todos los lugares donde aparezca la clase padre, es decir, la clase hija puede ser sustituida por la clase padre. Este concepto es definido también como **herencia**, mediante el cual la clase hija hereda las propiedades de la clase padre, especialmente los atributos y los métodos. Esta relación es muy común en clases que utilizan métodos que ya están implementados en otras clases.

- **Realización**: es un tipo de relación especial entre clases e interfaces, que indica que una clase realiza (implementa) los métodos definidos en la interface.

La representación gráfica que caracteriza cada relación entre las clases, se presenta más adelante en los diagramas de clases y responsabilidades de la sección 4.2.3.

4.2. Análisis de los mecanismos de colaboración

El análisis de los mecanismos de colaboración constituye la etapa inicial del desarrollo del espacio de trabajo compartido para Alliance. Como se ha mencionado, estos mecanismos están orientados a facilitar la negociación entre los autores durante la edición de un documento en forma colaborativa. Dichos mecanismos implican la necesidad de proporcionar el intercambio de mensajes entre los autores y promover la conciencia de grupo síncrona, mediante la difusión distribuida de los eventos que ocurren.

Este análisis se inicia con la definición de los mecanismos para la colaboración y la identificación de los diversos escenarios en los que interactúa cada autor, cuando se encuentra unido a la sesión de grupo. Posteriormente, se elaboran los diagramas de clases que componen el espacio de trabajo y se identifican las responsabilidades de las clases en cada escenario. Finalmente, se elaboran los diagramas de secuencia, los cuales determinan el comportamiento de cada clase en cada escenario y la forma como colaboran las diversas clases frente a la interacción de los autores.

4.2.1. Definición

Los mecanismos para la discusión y la votación, se basan en el establecimiento de una sesión de grupo para la comunicación y el intercambio de la información, de manera síncrona, entre los participantes. En [GrRo99][GaHa97][Kin97][Bur99][GuFu99], se define una **sesión** como el punto de reunión de diferentes entidades relacionadas para compartir información. Dichas entidades se definen como clientes, los cuales se unen a una sesión, para constituir el origen y el destino de la información que se intercambia dentro de ella.

En una sesión se establecen canales de comunicación, comunes para todos los clientes, a través de los cuales se transmite la información que se requiere compartir. Una vez se encuentran unidos a la sesión de grupo, los clientes se conectan (se agregan) a los canales y comparten la información mediante el envío y la recepción de mensajes a través de ellos. En términos generales, el envío del mensaje consiste en escribir su información en el canal, mientras que la recepción del mensaje, consiste en monitorear o consumir el contenido del canal.

Generalmente, para la difusión de mensajes sobre un canal de comunicación común, se asume una política optimista de entrega. Esta política se basa en el establecimiento de canales de tipo confiable, es decir, donde se garantiza la escritura del mensaje en el canal común y se garantiza la recepción del mensaje, en los clientes que han sido agregados al mismo. Esta política maneja un tiempo de espera (*"TimeOut"*), para verificar que cada cliente que consume el canal, haya recibido la señal de que puede obtener los mensajes enviados. En las secciones 4.3.2 y 4.4.2, se tratan las características del JSDT para el envío de mensajes sobre canales confiables.

El uso de canales comunes para la difusión de eventos y mensajes, permite que los autores reunidos en una sesión puedan discutir y realizar votaciones de manera síncrona. Una discusión síncrona es diferente a los hilos de discusión asíncrona en la Web, como los incorporados en sistemas como BSCW y COARSY, ya que los aportes se difunden y se perciben en tiempo real.

En las discusiones síncronas, los aportes pueden difundirse en forma síncrona débilmente acoplada, dependiendo de la acción voluntaria del autor que difunde dicho aporte. Por otra parte, la recepción de los aportes enviados puede realizase en forma síncrona fuertemente acoplada, para que los autores los reciban en el instante en que se envían y no intervengan con una acción voluntaria de recepción.

El proceso de votación presenta un comportamiento similar al de las discusiones síncronas, ya que al emitir un voto, el autor puede realizar la difusión en forma síncrona débilmente acoplada y al recibir el voto, dicha recepción puede llevarse a cabo de manera síncrona fuertemente acoplada. De acuerdo con estas características, a continuación se definen los mecanismos de colaboración y los aspectos para su análisis y diseño.

Discusión

Este mecanismo se refiere a que los autores puedan escribir sus ideas sobre temas específicos y compartirlas con los demás autores que trabajan en conjunto. Para ello, cada autor puede difundir sus diferentes aportes, mediante el envío de mensajes a través del canal de comunicación de la discusión, los cuales son recibidos en forma síncrona por el grupo de autores que participa en ella.

Debido a que los autores pueden unirse a una sesión de grupo, en diferentes instantes de tiempo, el mecanismo de discusión debe facilitar que cada autor se entere de las discusiones que se llevan a cabo en el momento de su ingreso a la sesión. Esta situación implica que los autores se enteren de todas las actividades que ocurren en la sesión, en el instante en que suceden, para mantener así un estado actualizado de la interacción que se lleva a cabo.

Adicionalmente, el mecanismo de discusión debe permitir que los mensajes intercambiados entre los participantes, puedan incorporarse al documento que está siendo editado en forma colaborativa. Asimismo, este mecanismo debe permitir que los autores almacenen los aportes de cada discusión.

Votación

Este mecanismo se refiere a que los autores pueden elegir y seleccionar las ideas, comentarios, sugerencias, etc., que desean aportar al documento editado. Esta elección puede presentarse también sobre la estructura del documento y sobre los roles atribuidos a cada autor para la edición.

Para ello, los autores deben disponer de un mecanismo que les permita formular una idea, difundirla entre el grupo de autores en la sesión y emitir su voto a favor o en contra de dicha idea en cuestión. El resultado de esta votación debe ser percibido en forma síncrona por los autores, para que incida en forma directa sobre sus contribuciones.

4.2.2. Casos de uso y escenarios

En UML, los **casos de uso** constituyen la forma para identificar y estructurar los requerimientos funcionales de un sistema. Los casos de uso establecen el comportamiento del sistema frente a estos requerimientos, mediante la identificación de diversos escenarios. Los **escenarios** comprenden una secuencia de acciones que reflejan un comportamiento, a partir del flujo de eventos que ocurre en forma normal y en forma excepcional.

Los casos de uso también permiten identificar los tipos de usuarios que interactúan con el sistema, representándolos como actores de los escenarios. Los actores utilizan el sistema e

interactúan con él mediante los casos de uso. Con base en estas características, se definen tres casos de uso para los mecanismos de colaboración descritos:

- **Establecer comunicación**. Inicialmente, los autores necesitan entablar comunicación directa entre sí, para poder discutir sus ideas o para promover una votación. Este caso de uso se refiere a las operaciones que permiten que un autor forme parte de la sesión de grupo y la forma en que se pueden enviar mensajes a los demás participantes.

- **Discusión**. Se refiere a las operaciones del proceso de generar discusiones sobre temas determinados, la forma en que los autores pueden contribuir en ellas y la percepción de dichas contribuciones en el instante en que suceden. Adicionalmente, este caso de uso se refiere al proceso que le permite a un autor unirse a una discusión que se lleva a cabo, en el momento en que ingresa a la sesión.

- **Votación**. Se refiere a las operaciones relacionadas con el proceso de someter una idea a la opinión de los autores en la sesión. Además, este caso de uso se refiere al proceso que permite a cada autor emitir su voto y enterarse del resultado de la votación en forma inmediata.

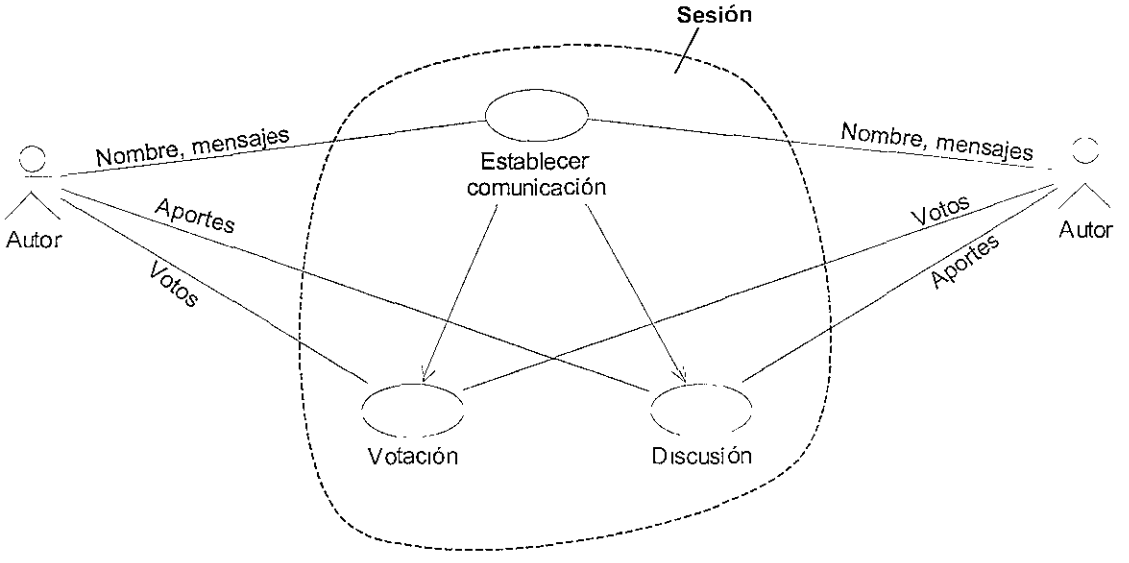


Figura 4.1. Diagrama de casos de uso del espacio de trabajo compartido.

La figura 4.1 presenta el diagrama de los casos de uso que conforman una sesión del espacio de trabajo compartido. Nótese que el autor está identificado como el único actor que interactúa con el espacio de trabajo. Nótese también que los autores establecen comunicación entre sí antes de realizar sus aportes a las discusiones o emitir sus votos. Asimismo, cuando los autores se encuentran reunidos en la sesión pueden comunicarse mediante el intercambio de mensajes.

En cada caso de uso se identifican diversos escenarios, en los cuales los autores realizan acciones específicas requeridas para su interacción. La definición de estos escenarios, así como sus flujos normales y excepcionales, se presentan en las siguientes tablas:

Caso de uso	Escenario	Flujo normal	Flujo excepcional
Establecer comunicación	Unirse a la sesión	El sistema solicita el nombre del autor que desea unirse a la sesión. El autor se identifica como un nuevo cliente para ser ingresado a la sesión. El sistema valida la existencia del autor dentro de la sesión. Si el autor no existe en la sesión, el sistema lo agrega (conecta) como cliente del canal de comunicación establecido para ella El autor unido a la sesión recibe la lista de sus demás colegas en la sesión y la lista de las discusiones que se encuentran en proceso	El autor ya existe en la sesión, el sistema no puede agregar el autor al canal de la sesión porque está desconectado de la red de comunicación
	Enviar mensaje	El autor escribe un mensaje para difundirlo a uno o a todos los autores que interactúan en la sesión. El sistema identifica dicho evento de envío y escribe el mensaje en el canal de comunicación establecido. El sistema recibe la notificación de que pudo escribir el mensaje en el canal de comunicación Si dicha notificación no es recibida, el sistema cancela la operación	El autor no puede enviar el mensaje porque está desconectado de la red de comunicación

Tabla 4.1. Escenarios del caso de uso establecer comunicación.

Como se presenta en la tabla 4.1, los escenarios definidos para el caso de uso establecer comunicación comprenden **unirse a la sesión** y **enviar mensaje**. Una vez que el autor se encuentra unido a la sesión de grupo se comunica con los demás autores mediante la difusión de mensajes. El espacio de trabajo utiliza una política optimista para la difusión de mensajes en forma síncrona sobre canales de comunicación confiables. Debido a esto, cuando no se puede enviar el mensaje por el canal establecido para la sesión se opta por cancelar la operación.

Caso de uso	Escenario	Flujo normal	Flujo excepcional
Votación	Promover votación	El autor promueve la votación, escribiendo una idea sobre la cual debe elegirse la aprobación o el desacuerdo. El sistema identifica el evento de votación, crea un canal de comunicacion para ésta y agrega al autor como cliente de dicho canal. Si no se pudo establecer el canal para la votación el sistema cancela la operación. Una vez creado el canal, el sistema difunde la idea sometida a los demás autores de la sesión. Para difundir la idea sometida a votación, el sistema utiliza el canal establecido para la sesión El sistema agrega a cada autor que recibe la idea sometida, como cliente del canal de la votación. Cada autor puede promover una votación en cualquier momento y a su vez, el sistema le permite almacenar el resultado de cada votación	El autor no puede promover la votación porque está desconectado de la red de comunicación
	Realizar votación	El autor emite su voto sobre la idea difundida y el sistema deshabilita su opción de votar El sistema identifica el evento de voto emitido y escribe el mensaje en el canal de comunicación de la votación. El sistema recibe la notificación de que pudo escribir el mensaje en el canal de comunicación Si dicha notificación no es recibida, el sistema cancela la operación Una vez difundido el voto emitido, el sistema contabiliza los votos y difunde el resultado de la votación a los participantes de la sesión	El autor no puede emitir su voto porque está desconectado de la red de comunicación

Tabla 4.2. Escenarios del caso de uso votación.

En la tabla 4.2 se presentan los escenarios descritos para el caso de uso votación: **promover votación** y **realizar votación**. Una vez que el autor se encuentra activo en la sesión, puede promover votaciones y participar en ellas, difundiendo su voto a todos los participantes de la sesión y obteniendo el resultado actualizado de cada votación. Para cada votación promovida se establece un canal independiente para que las votaciones sobre ideas diferentes puedan ocurrir al mismo tiempo.

Caso de uso	Escenario	Flujo normal	Flujo excepcional
Discusión	Generar discusión	El autor genera una discusión escribiendo el tema sobre el cual se va a discutir El sistema identifica el evento de discusión, crea un canal de comunicación para la discusión y agrega al autor como cliente de dicho canal Si no se pudo establecer el canal para la discusión el sistema cancela la operación Una vez creado el canal, el sistema actualiza el registro de las discusiones que han sido generadas. Posteriormente el sistema difunde el tema de la discusión a los demás autores en la sesión, para que decidan si van a participar en ella. Para la difusión del tema de la discusión, el sistema utiliza el canal establecido para la sesión El sistema agrega a cada autor que decide participar en la discusión, como cliente del canal establecido para ella. Asimismo, el sistema actualiza la lista de participantes de la discusión y la difunde entre ellos. Cualquier autor puede generar una discusión en cualquier momento y a su vez, el sistema le permite almacenar los aportes de cada discusión	El autor no puede generar la discusión porque está desconectado de la red de comunicación
	Unirse a una discusión	El autor selecciona la discusión a la que se desea unir, consultando la lista de discusiones de la sesión El sistema agrega el autor como cliente del canal establecido para la discusión seleccionada Si el sistema no pudo agregar el autor al canal, cancela la operación. Si se pudo agregar el autor al canal, el sistema lo agrega a la lista de participantes de la discusión y difunde dicha lista entre ellos	El autor no puede unirse a la discusión porque está desconectado de la red de comunicación
	Agregar aporte a la discusión	El autor escribe su aporte a la discusión El sistema identifica el evento de difusión de dicho aporte y escribe el mensaje en el canal de comunicación de la discusión El sistema recibe la notificación de que pudo escribir el mensaje en el canal de comunicación. Si dicha notificación no es recibida, el sistema cancela la operación	El autor no puede agregar su aporte a la discusión porque está desconectado de la red de comunicación

Tabla 4.3. Escenarios del caso de uso discusión.

La tabla 4.3 presenta la descripción de los escenarios **generar discusión**, **unirse a una discusión** y **agregar aporte a la discusión**, del caso de uso discusión. Una vez que el autor se encuentra activo en la sesión, puede generar nuevas discusiones, puede aceptar su participación en las discusiones generadas o puede unirse a las discusiones que ya se llevan a cabo. A diferencia de las votaciones, los autores pueden elegir las discusiones en las que desean participar.

4.2.3. Clases y responsabilidades

Para cada escenario de cada caso de uso definido, se establecen las clases que determinan el comportamiento del sistema frente a la interacción de cada autor. Para ello, es necesario identificar las clases de propósito especifico, que provee tanto el API JSDT como el lenguaje Java, las cuales son utilizadas como la base para la construcción del espacio de trabajo.

Existen tres tipos de clases: las que se identifican como domino del problema, es decir, directamente del comportamiento que debe tener el sistema, las que proporciona el lenguaje Java para generar el ambiente gráfico para los autores y las que proporciona el API JSDT para funciones como la creación de sesiones de grupo y el establecimiento de canales de comunicación. Estos tipos de clases se definen como sigue:

Clases del domino del problema

- **Discusión**: es la clase que identifica cada tema que se somete a discusión. Esta clase define una interface gráfica de ventana para cada autor que participa en la discusión, en la que se presentan los mensajes aportados por cada uno y la lista de los participantes.

- **Votación**: es la clase que identifica cada idea que se somete a votación. Esta clase define una interface gráfica de ventana para cada autor, en la que se presentan las opiniones de acuerdo y de desacuerdo, que cada uno emite cuando se promueve una votación.

- **Usuario**: es la clase que define la interface gráfica de ventana principal del sistema, es decir, el espacio de trabajo compartido. Esta clase involucra la representación de la información de la sesión y de las operaciones que el autor puede realizar dentro del espacio de trabajo.

Clases de la interface gráfica

- **Java.awt.event**: es la clase utilizada para el manejo de los eventos que ocurren en todas las ventanas que componen el espacio de trabajo compartido.

- **Java.awt.Frame**: es la clase que construye las ventanas y los demás elementos que componen cada interface gráfica del sistema (menús, cuadros de texto, listas, etc.).

Interfaces y clases del soporte para la comunicación

- **Com.sun.media.jsdt.Client**: es la interface que representa a cada autor que participa en la sesión de grupo. Esta interface se requiere para que el autor pueda ser unido a los canales de comunicación y a la sesión.

- **Com.sun.media.jsdt.ChannelConsumer**: es la interface que utilizan los clientes para recibir los mensajes. Esta interface declara (no implementa) el método **dataRecieved()**, el cual se encarga de notificar la existencia de un mensaje en el canal. Debido a que ChannelConsumer es una interface, las operaciones que se deben llevar a cabo cuando se consumen los mensajes, deben ser indicadas en forma explícita para cada consumidor del canal. Cada canal tiene diversos tipos de consumidores, los cuales manipulan de diferentes formas los mensajes que reciben.

- **Com.sun.media.jsdt.Session**: es la clase que define el punto de reunión del grupo de autores (clientes). Esta clase se encarga también de establecer los canales de comunicación requeridos en la sesión. Además, esta clase utiliza la clase **com.sun.media.jsdt.SessionFactory**, la cual se encarga de crear la sesión, de acuerdo a una serie de parámetros establecidos, como por ejemplo el protocolo de comunicación utilizado. En las secciones 4.3.2. y 4.4.2 se explica la definición de estos parámetros para la creación de una sesión JSDT.

- **Com.sun.media.jsdt.Channel**: es la clase que identifica cada canal de comunicación creado en la sesión del espacio de trabajo, a través de los cuales se comparte la información entre los clientes. Esta clase también se encarga de agregar a los clientes como consumidores de cada canal.

De acuerdo con la definición de interface (ver sección 4.1), para implementar las operaciones de los métodos que declara, es necesario establecer una clase, es decir, la clase realiza los métodos que declara la interface. Por ello, para las interfaces Client y ChannelConsumer,

utilizadas en el espacio de trabajo compartido, se establecen las clases **Cliente** y **Consumidor** respectivamente. Para el caso de las clases Session y Channel, se establecen los nombres **Sesión** y **Canal** respectivamente, para conservar nombres homogéneos.

El comportamiento de cada clase y las relaciones existentes entre ellas se especifica utilizando el diagrama estructural de clases [Boo99]. En este diagrama se establecen las responsabilidades de cada clase y la forma como cada una colabora (se relaciona) con las demás. Las responsabilidades se refieren a la descripción textual de las funciones que realiza cada clase, con el fin de identificar los métodos que implementan su comportamiento.

En el análisis del comportamiento de un sistema, usualmente se realiza un solo diagrama de clases general, donde se representan las diferentes relaciones que existen entre todas las clases, en todos los casos de uso. En la presente investigación, se realiza un diagrama de clases y responsabilidades por cada escenario de cada caso de uso, para facilitar su entendimiento y su diseño. Cada escenario muestra sólo una parte del comportamiento del sistema, por lo que la suma de las responsabilidades de cada clase en cada escenario, corresponde al total de las responsabilidades de cada clase en el sistema completo. A continuación se presentan los diagramas de clases y las responsabilidades.

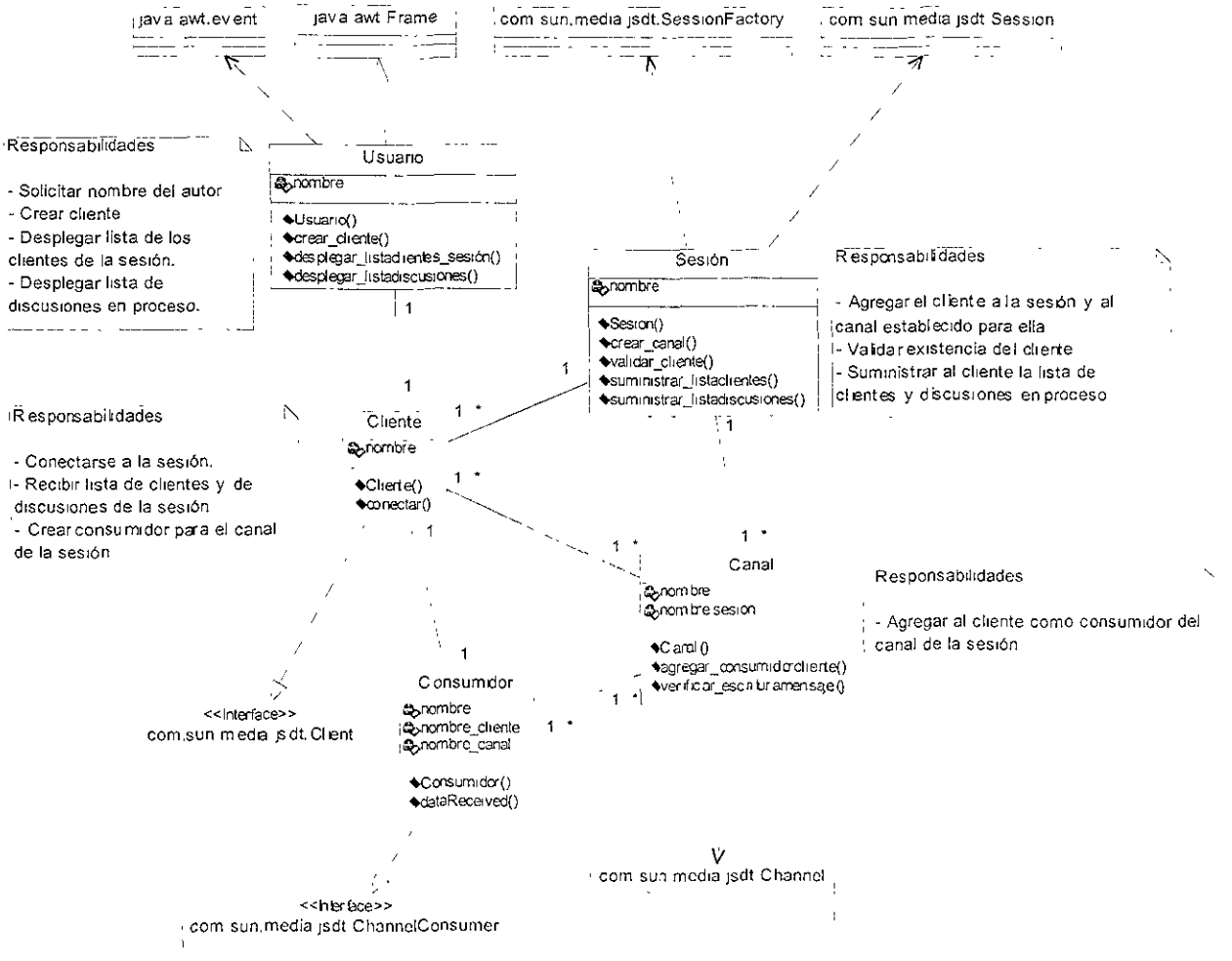


Figura 4.2. Diagrama de clases y responsabilidades del escenario unirse a la sesión.

En la figura 4.2 se presenta el diagrama de clases y responsabilidades del escenario unirse a la sesión. Cada clase tiene asociada sus atributos (recuadro central de cada clase) y sus métodos (recuadro inferior de cada clase). Por ejemplo, la clase Usuario solamente tiene como atributo el **nombre** y los métodos definidos en este escenario, son: **Usuario()**, **crear_cliente()**, **desplegar_listaclientes_sesión()** y **desplegar_listadiscusiones()**.

Nótese que para cada clase existe un método del mismo nombre, por ejemplo Sesión(), Usuario(), Consumidor(), etc. Estos métodos se denominan constructores de la clase y su función es crear objetos, asignándoles los valores para sus atributos. De esta forma, las responsabilidades de cada clase respecto a su propia creación, quedan indicadas con los métodos constructores. Nótese también, que los demás métodos pueden deducirse de las responsabilidades establecidas para cada clase.

En este diagrama se introduce la asociación existente entre las clases del JSDT (Sesión, Cliente, Canal y Consumidor) y las clases del dominio del problema, que en este escenario sólo incluye la clase Usuario. Para cada autor, debe existir una clase Usuario, la cual despliega la interface gráfica del sistema, y una clase Cliente, mediante la cual se recibe la información de la Sesión, del Canal y del Consumidor del Canal. Nótese que la clase Usuario es la que crea la clase Cliente, para permitir que dicha información recibida puede ser desplegada al autor.

Es importante resaltar que la multiplicidad de las asociaciones presentadas en este diagrama hace referencia a que existen diversos autores que interactúan en la sesión. Por ello, dicha multiplicidad se define en los siguientes casos:

- Cada usuario crea su objeto Cliente para identificarse dentro la sesión (multiplicidad 1 1).
- A su vez, cada objeto de la clase Cliente puede conectarse a diferentes objetos de la clase Canal y cada canal agrega uno o más clientes (multiplicidad 1..* 1..*).
- Cada cliente crea un objeto Consumidor del canal (multiplicidad 1 1), con el cual recibe la información que se envía a través de él.
- Cada objeto de la clase Canal está asociado a un solo objeto de la clase Sesión (multiplicidad 1..* 1), puesto que todos los clientes se reúnen en una sola sesión de grupo.

En el diagrama, también se presentan las relaciones de dependencia entre las clases: Usuario que depende de java.awt.event, Sesión que depende de com.sun.media.jsdt.SessionFactory y com.sun.media.jsdt.Session, y Canal que depende de com.sun.media.jsdt.Channel. Asimismo, la generalización o herencia se presenta en la clase Usuario, la cual hereda los métodos y atributos de la clase java.awt.Frame.

Por último, la relación de realización se presenta con las clases Cliente y Consumidor, las cuales implementan los métodos declarados en las interfaces com.sun.media.jsdt.Client y com.sun.media.jsdt.ChannelConsumer, respectivamente.

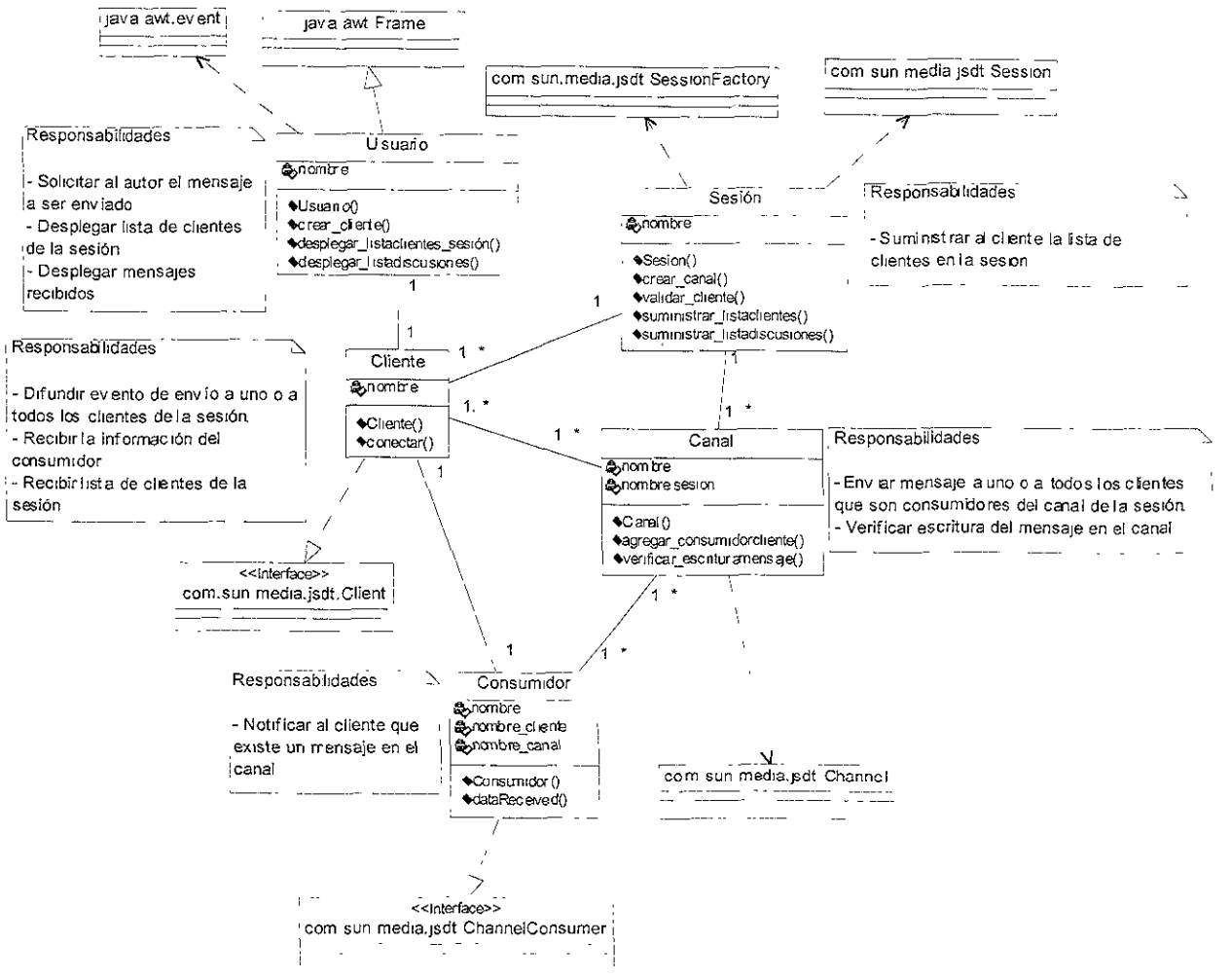


Figura 4.3. Diagrama de clases y responsabilidades del escenario enviar mensaje.

En la figura 4.3 se presentan las responsabilidades de cada clase identificada en el escenario enviar mensaje. Nótese que en este diagrama no se presenta ninguna clase adicional a las de la figura 4.2, debido a que la operación de envío del mensaje involucra directamente a las clases que ya se han identificado.

Dicha operación se resume como sigue: la clase Usuario se encarga de solicitar el mensaje a un autor y desplegarlo a los demás, la clase Cliente se encarga de difundir el evento de envío a uno o a todos los autores, la clase Canal se encarga de escribir el mensaje en el canal común para todos los clientes y la clase Consumidor se encarga de notificar que el mensaje está disponible para su respectivo cliente.

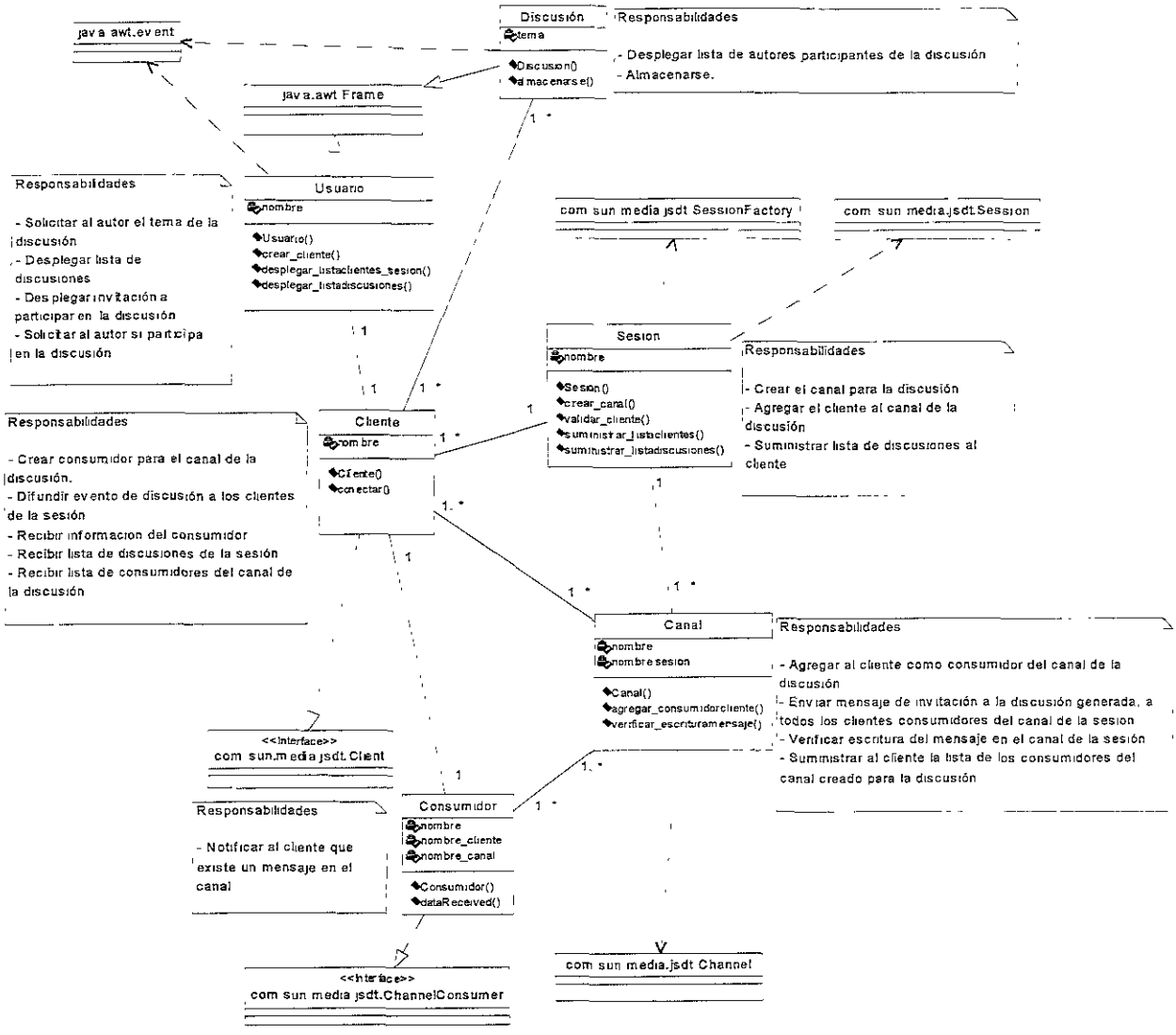


Figura 4.4. Diagrama de clases y responsabilidades del escenario generar discusión.

La figura 4.4 presenta el diagrama de clases y responsabilidades del escenario generar discusión. Nótese que en este diagrama se agrega la clase Discusión, la cual se asocia directamente con la clase Cliente, puesto que cada discusión involucra varios clientes y cada cliente puede participar en varias discusiones (multiplicidad 1..* 1..*).

Como se mencionó anteriormente, la clase Discusión se encarga de desplegar la interface gráfica de ventana para cada tema sobre el cual se discute y los aportes de cada autor. De igual forma, esta clase despliega la lista de los participantes que aceptan la invitación difundida por el cliente que genera la discusión.

En este diagrama se muestra que la clase Cliente desempeña funciones tanto para el autor que genera la discusión, como para los demás autores que reciben el evento de invitación a participar en ella. Asimismo, la clase Sesión se encarga de crear el canal de comunicación, diferente e independiente para cada discusión, y agregarle los respectivos clientes que aceptan su participación.

Nótese que tanto para el cliente (autor) que genera la discusión como para los clientes (autores) que deciden participar en ella, existe un objeto de la clase Consumidor. Dichos objetos son utilizados por la clase Canal, para agregar a sus respectivos clientes como consumidores del canal creado para cada discusión. De esta forma, los clientes se agrupan en cada discusión y pueden enterarse de los eventos que suceden en ella, por medio de la información difundida y compartida en el respectivo canal.

En este diagrama se muestra también, que la clase Usuario se encarga de solicitar al autor el tema para generar la discusión y desplegar a los demás autores, la invitación a participar en ella. En el momento en que el autor decide participar en la discusión, la clase Usuario crea la clase Discusión (interface gráfica de la discusión) para ese autor. Sin embargo, aunque la clase Usuario crea la clase Discusión, no es necesario establecer una asociación directa entre ellas, ya que son solamente interfaces gráficas y la clase Cliente es la que representa al autor en cada discusión en la que participa.

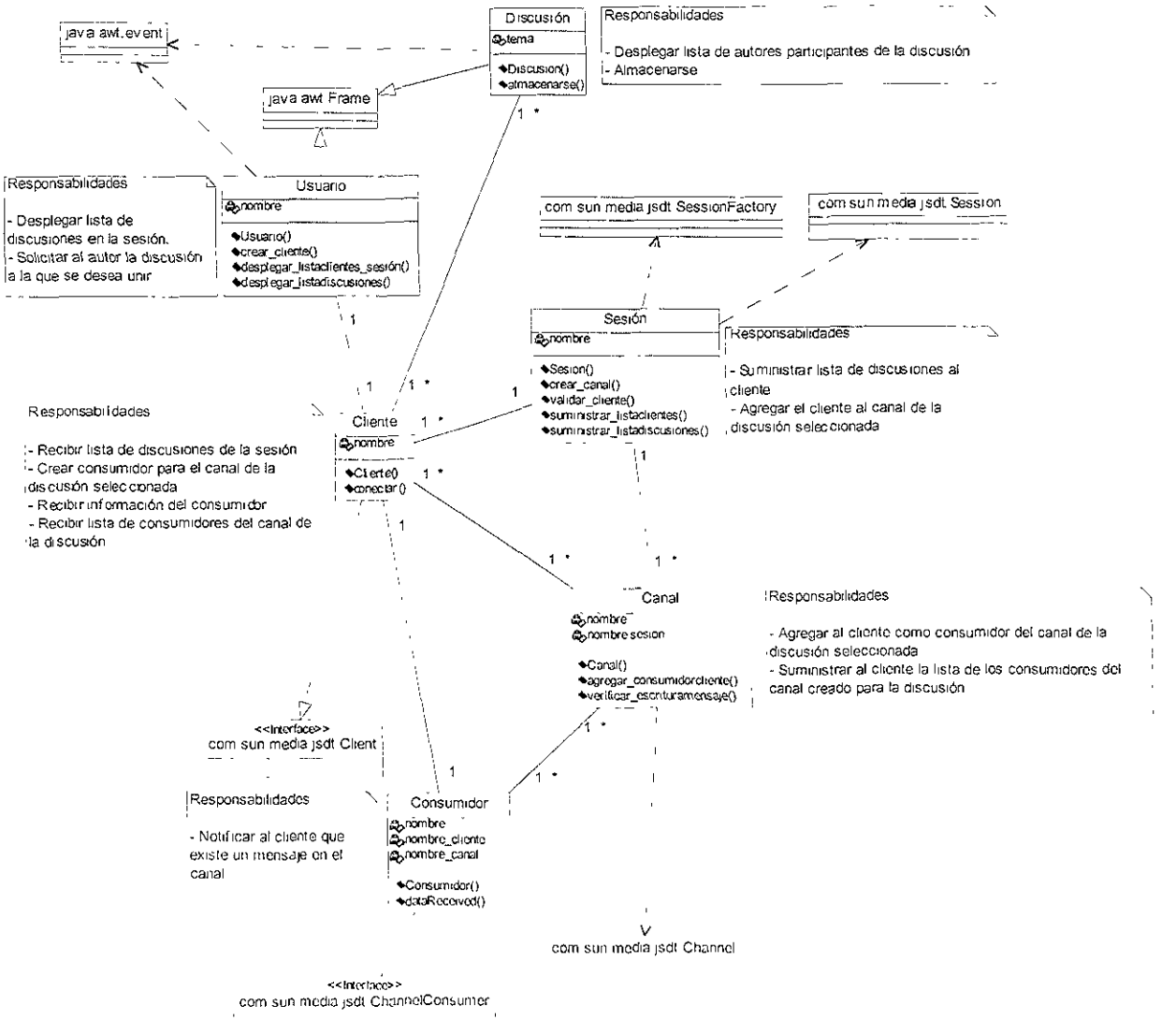


Figura 4.5. Diagrama de clases y responsabilidades del escenario unirse a una discusión.

La figura 4.5 presenta las responsabilidades de las clases correspondientes al escenario unirse a una discusión, el cual ocurre cuando un autor ingresa a la sesión del espacio de trabajo, en el momento en que existen discusiones en proceso.

Este diagrama muestra que las responsabilidades de las clases Usuario, Cliente, Discusión y Consumidor, se llevan a cabo sólo para el autor que ingresa a la discusión que ha seleccionado. Sin embargo, el hecho de que el autor se agregue como consumidor del canal de la discusión, produce la actualización de la lista de consumidores de dicho canal, lo cual debe reflejarse en la interface gráfica de los demás autores que ya participan en ella.

Esta situación no se muestra en este diagrama debido a que los autores que ya participan en la discusión, fueron agregados al canal creado para ella en el escenario generar discusión, y por lo mismo, pueden recibir de dicho canal los eventos y desplegarlos en sus respectivas interfaces de la discusión.

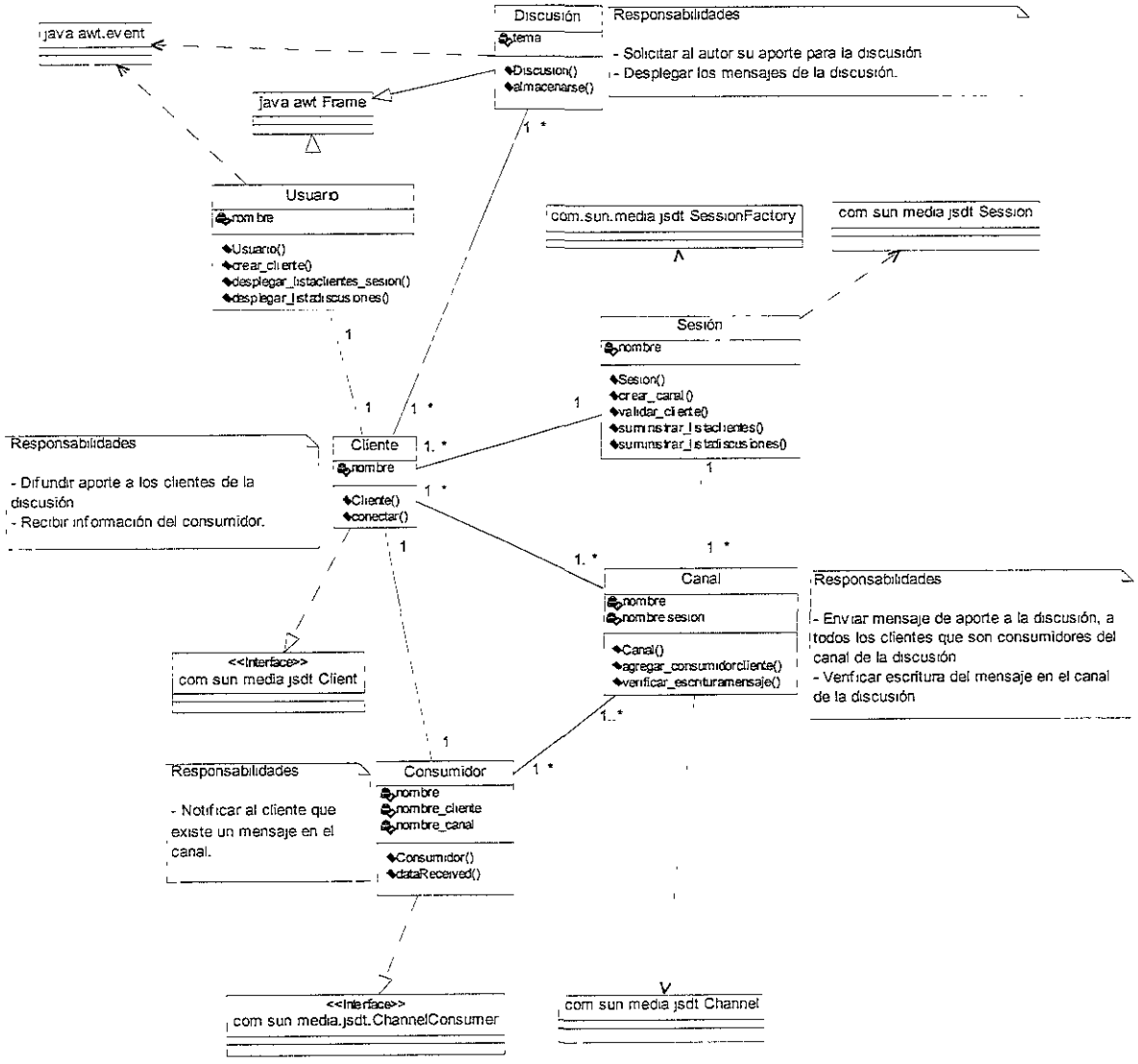


Figura 4.6. Diagrama de clases y responsabilidades del escenario agregar aporte a la discusión.

En la figura 4.6 se presenta el diagrama de clases y responsabilidades del escenario agregar aporte a la discusión. Nótese que las responsabilidades atribuidas a las clases Discusión, Cliente y Consumidor, se llevan a cabo para cada autor que participa en la discusión.

En este diagrama se muestra que una vez que se agrupan los autores en la discusión, la interacción se lleva a cabo solamente entre las clases Discusión, Cliente, Consumidor y Canal, por lo que las clases Sesión y Usuario no desempeñan ninguna responsabilidad en este escenario.

El proceso de difundir un aporte a la discusión, se realiza mediante la escritura de un mensaje apropiado en el canal establecido para ella. Esto es, una vez que el autor ingresa su aporte en la interface de la discusión, su respectivo cliente difunde este evento, a través de un mensaje que se escribe en el canal. Posteriormente, los consumidores agregados como clientes de dicho canal, realizan la función de recepción del mensaje para desplegarlo en sus correspondientes interfaces. Esta última operación se facilita porque la clase Cliente posee una asociación directa con la clase Discusión.

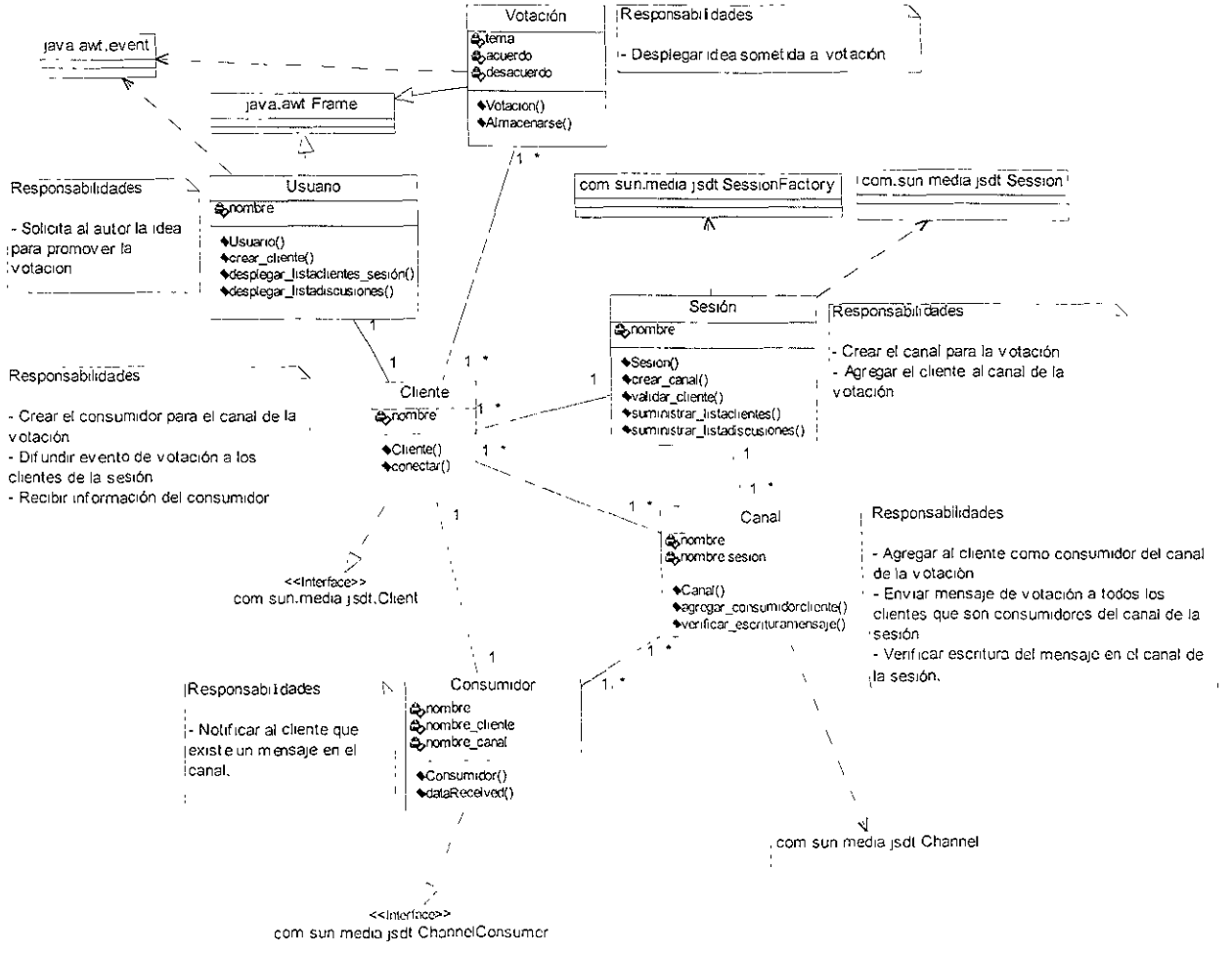


Figura 4.7. Diagrama de clases y responsabilidades del escenario promover votación.

En la figura 4.7 se presentan las responsabilidades de las clases identificadas para el escenario promover votación. En dicho diagrama se agrega la clase Votación, la cual se asocia directamente con la clase Cliente, porque un cliente puede participar en varias votaciones y las votaciones se llevan a cabo con la participación de varios clientes (multiplicidad 1..* 1..*).

En este diagrama se presentan responsabilidades que son muy similares a las que se llevan a cabo en el escenario generar discusión, por ejemplo, la clase Sesión se encarga de crear el canal de comunicación para cada votación. De igual forma, en este diagrama no se presenta una relación directa entre la clase Votación y la clase Usuario, puesto que son clases que corresponden al despliegue de las interfaces gráficas y la clase Cliente es la que realmente está asociada con la clase Votación.

A diferencia del evento que ocurre una vez generada la discusión, cuando se promueve una votación no se difunde una invitación a participar en ella. Esto quiere decir, que el escenario promover votación involucra siempre la participación de los autores que se encuentran en la sesión, ya que se descartan aquellos autores que se unen a la sesión luego de que se ha promovido la votación.

Por esta razón, el evento de promover la votación ocasiona que en cada cliente se defina un objeto de la clase Votación, para desplegar la idea sometida que se recibe por medio del canal de la sesión. Como paso previo a la creación de la interface gráfica de la votación (objeto de la clase Votación), cada cliente es agregado al canal creado para la votación promovida.

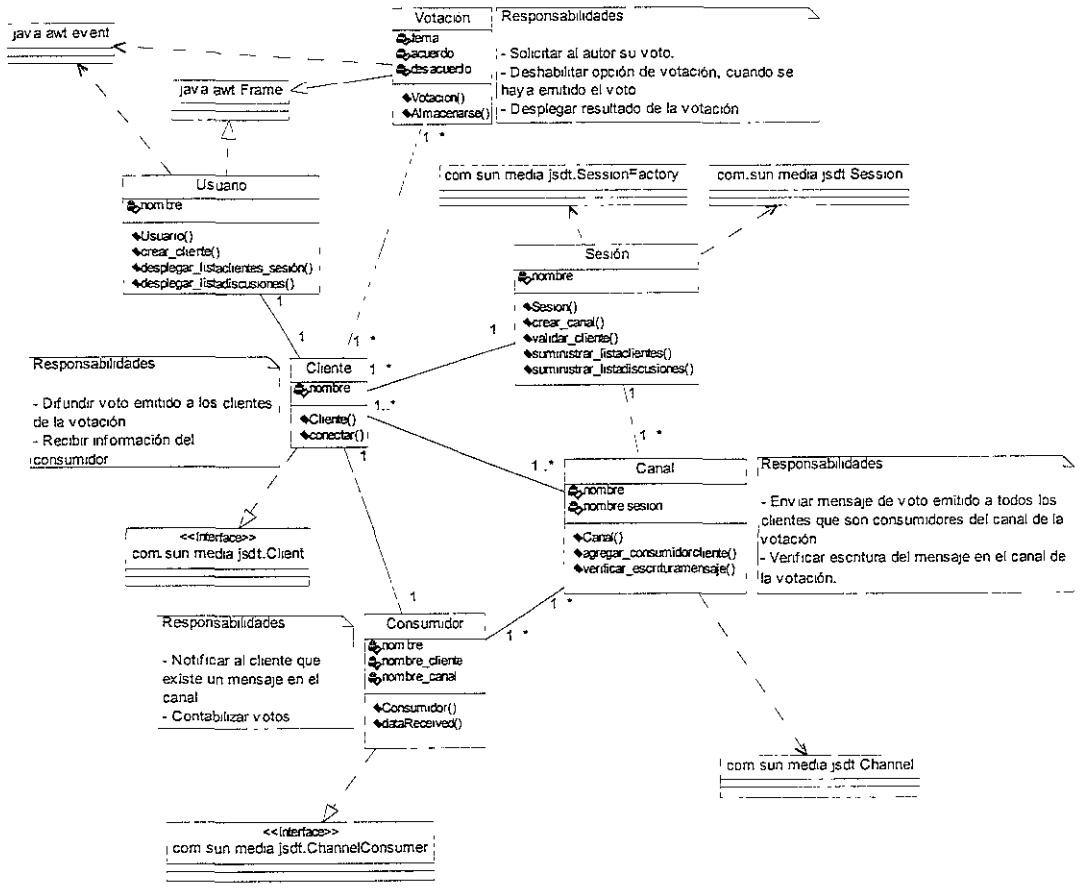


Figura 4.8. Diagrama de clases y responsabilidades del escenario realizar votación.

En la figura 4.8 se presenta el diagrama de clases y responsabilidades del escenario realizar votación, el cual ocurre una vez se ha difundido a cada autor la idea sometida. Al igual que en el escenario agregar aporte a la discusión, las clases Usuario y Sesión no desempeñan ninguna responsabilidad en este diagrama, es decir el proceso de realizar la votación se lleva a cabo solamente con las clases Votación, Cliente, Canal y Consumidor.

Como se ha explicado, para cada votación se utiliza un canal diferente al canal de la sesión, porque la información que se comparte de cada votación, es diferente a la información de los eventos que se difunden en la sesión. Este concepto se retoma en la sección 4.4.3, donde se especifican los canales del espacio de trabajo y los diferentes tipos de consumidores.

Finalmente, nótese que la operación de contabilizar los votos la realiza cada consumidor y no cada cliente. Esto sucede porque para un autor (cliente) no es relevante conocer el voto que emite cada uno de sus colegas, sino el resultado por cada opción (acuerdo o desacuerdo) que se va actualizando con cada mensaje que recibe el consumidor del canal.

4.2.4. Diagramas de secuencia (Comportamiento)

A partir de las clases y sus responsabilidades identificadas en cada escenario del sistema, se establece la forma como cada clase realiza sus operaciones frente a la interacción de los actores. Para ello, se utiliza el diagrama de secuencia [Boo99], en el cual se representa el comportamiento que tiene cada clase y la forma como interactúa con otras clases y con los actores.

En este diagrama se especifica el conjunto de objetos (instancias de las clases) que participan en la interacción y los mensajes que se transmiten entre ellos. En [Boo99] se define el diagrama de secuencia, como una tabla que presenta los objetos organizados como columnas y los mensajes organizados como filas, ordenados en tiempo ascendente. Los mensajes corresponden a la especificación lógica de una comunicación entre los objetos, la cual incluye la información suficiente para la realización de una acción. En UML se definen 5 tipos de acciones:

- **Llamar**: invocar una operación en un objeto.
- **Retornar**: regresar un valor a una llamada.
- **Enviar**: enviar una señal a un objeto.
- **Crear**: crear un objeto. Lo cual significa que se crea una instancia de una clase.
- **Destruir**: destruir un objeto y liberar los recursos que utiliza (memoria, conexiones, etc.).

Al igual que para los diagramas de clases, los diagramas de secuencias pueden simplificarse, representando la interacción que ocurre en cada escenario del sistema. Para reducir el número de mensajes en estos diagramas, se omite el proceso de establecer las clases para las interfaces del JSDT, esto es, la interacción que conlleva el crear la clase Consumidor y la clase Cliente, a partir de las interfaces Channel y Client respectivamente.

De igual forma, la interacción que conlleva el escribir un mensaje en un canal y esperar las señales de recepción de todos los consumidores, se simplifica por medio de la llamada **escribir_mensaje()**. Si no se reciben todas las notificaciones de los consumidores, se realiza la llamada **notificar_error()**, inmediatamente después de la llamada de escritura, para indicar que esta operación no pudo realizarse.

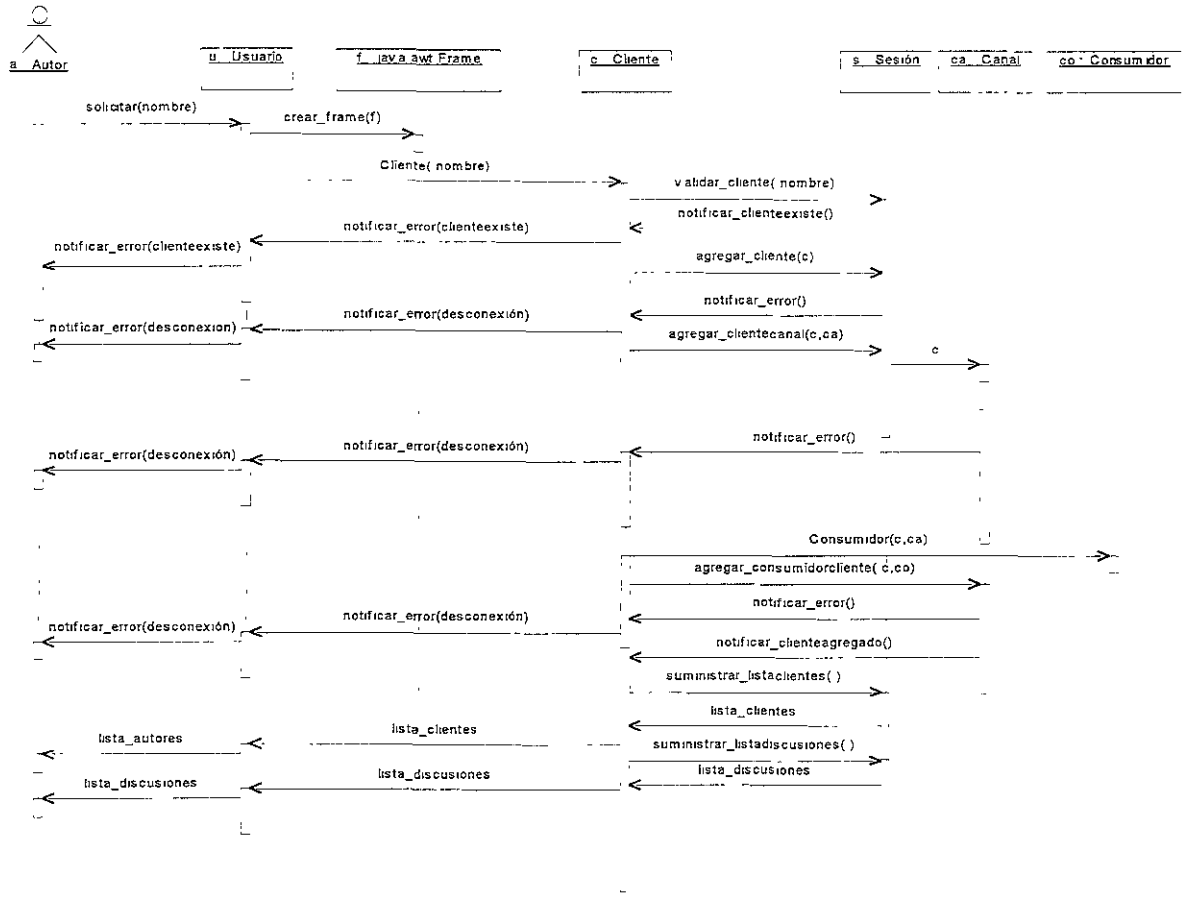


Figura 4.9. Diagrama de secuencia del escenario unirse a la sesión.

En la figura 4.9 se presenta la interacción establecida para el escenario unirse a la sesión. En este diagrama se introduce la forma de representar los objetos creados para cada clase, por ejemplo: **a** de la clase **Autor**, **u** de la clase **Usuario**, etc., ubicados como columnas. Nótese que en el diagrama se muestra que la clase Usuario crea un objeto **f** de la clase **java.awt.Frame**, para ejemplificar la construcción de la interface gráfica para el autor.

Para los mensajes que ocurren entre los objetos, solamente se utilizan tres de los cinco tipos de acciones: llamar, retornar y crear. Una llamada se representa como un mensaje con el nombre de un método, dirigido hacia el objeto que lo implementa. Por ejemplo, el autor llama el método **solicitar(nombre)**, el cual lo ejecuta la clase Usuario. Esta llamada significa que la clase Usuario solicita el nombre del autor para empezar la interacción en el escenario.

Las llamadas se utilizan también para ejecutar métodos en respuesta a una acción previa. Por ejemplo, cuando el objeto **c** llama el método **validar_cliente(nombre)** del objeto **s**, dicho objeto **s** devuelve otra llamada al método **notificar_clienteexiste()** del objeto **c**, para controlar que el cliente que se está ingresando ya existe en la sesión.

Los mensajes que indican un valor de retorno se representan con el nombre del valor de retorno[1] , dirigido hacia el objeto que recibe dicho valor. Por ejemplo, cuando el objeto **c** llama

[1] No se usan paréntesis para diferenciarlos de los nombres de los métodos.

el método **suministrar_listaclientes()** del objeto **s,** dicho objeto **s** retorna el valor **lista_clientes.** Los mensajes referentes a la creación de objetos se representan como llamadas al método constructor de la clase que los especifica. Por ejemplo, el objeto **u** crea el objeto **c**, llamando el constructor **Cliente(nombre).**

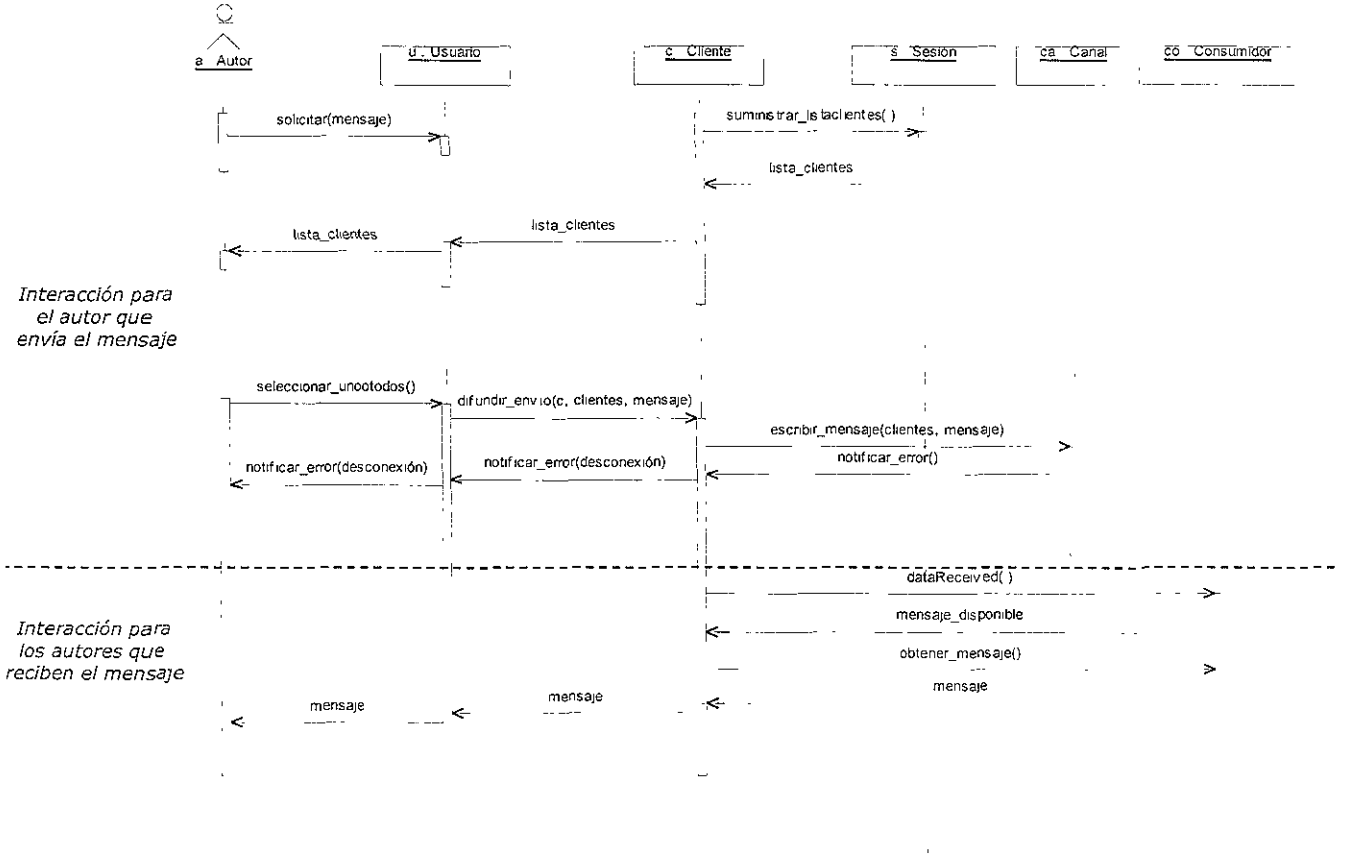


Figura 4.10. Diagrama de secuencia del escenario enviar mensaje.

La figura 4.10 presenta la secuencia del escenario enviar mensaje. En el diagrama se introduce la forma de representar una situación particular: el actor **autor** no sólo representa el autor que envía el mensaje, sino también todos los autores que lo reciben. Por ello, el diagrama se divide en dos partes, donde la parte superior corresponde a la interacción en la que participa el autor que envía el mensaje y la parte inferior corresponde a la interacción de los autores que lo reciben.

De acuerdo con la secuencia del escenario **unirse a la sesión**, cada cliente es agregado como consumidor del canal **ca** (de la sesión) en el momento en que ingresa a la sesión. Por ello, en el escenario **enviar mensaje**, cuando el autor genera la escritura en el canal, los clientes de los demás autores llaman el método **dataReceived()** y posteriormente procesan el mensaje obtenido para desplegarlo en sus interfaces.

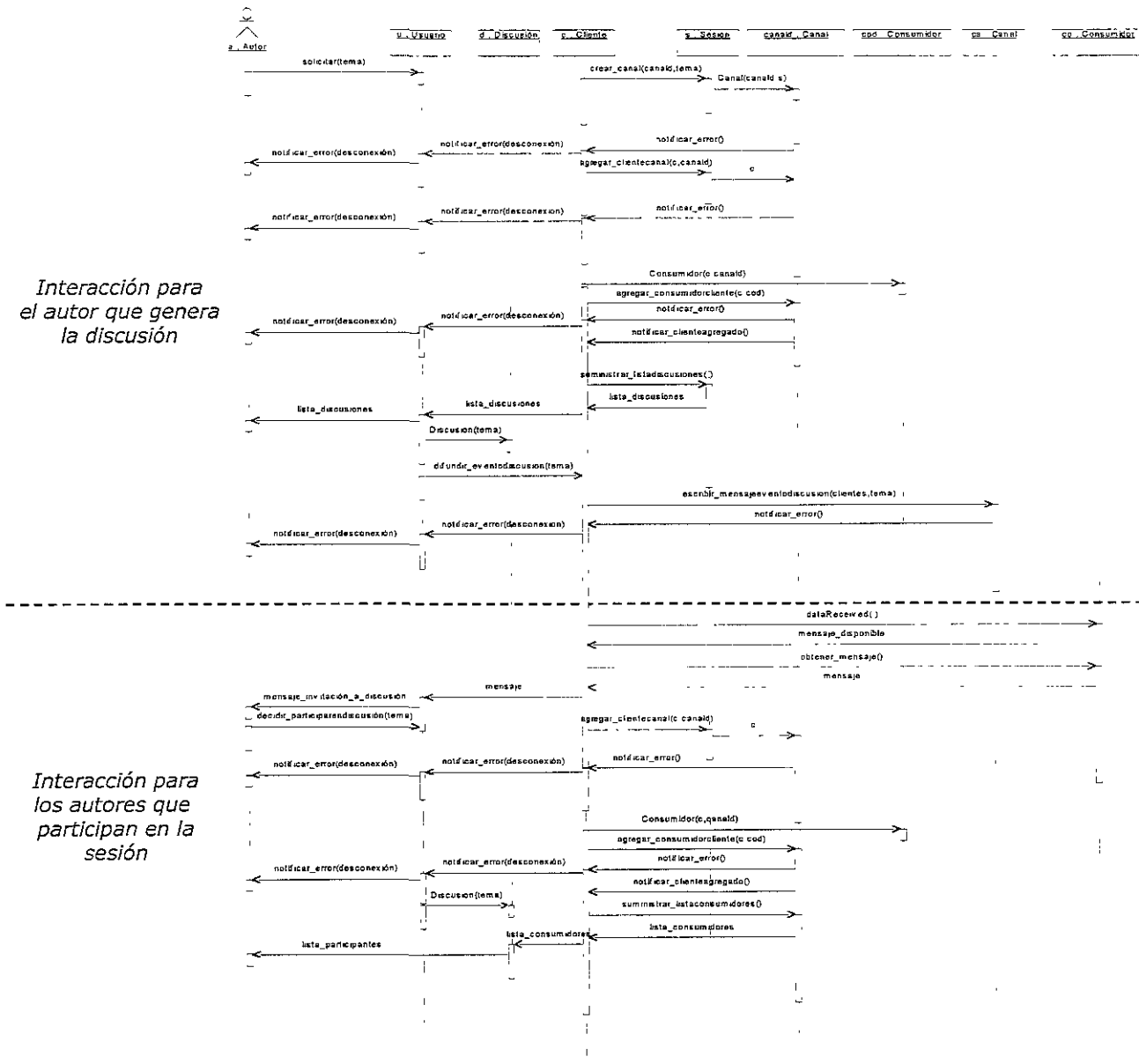


Figura 4.11. Diagrama de secuencia del escenario generar discusión.

En la figura 4.11 se presenta la secuencia que ocurre en el escenario generar discusión. En este diagrama se muestran dos objetos de la clase Canal, **ca** y **canald**, y dos objetos de la clase Consumidor **co** y **cod**. Los objetos **ca** y **co**, corresponden al canal de la sesión y al consumidor creado para dicho canal. Los objetos **canald** y **cod**, corresponden al canal creado para la discusión y el consumidor para este canal.

Para cada discusión se crea un canal independiente del canal de la sesión, porque en cada discusión se difunden eventos diferentes a los que se difunden para la sesión. Por ejemplo, la invitación a participar en la discusión, se difunde a todos los clientes del canal **ca**, mientras que la lista de participantes de cada discusión se difunde sólo a los clientes del canal **canald**.

Nótese que cuando un autor decide participar en la discusión generada, es agregado como cliente del canal de la misma. Posteriormente, el objeto **u** de cada autor, crea el objeto **d** (clase Discusión) para desplegar la lista de participantes actualizada.

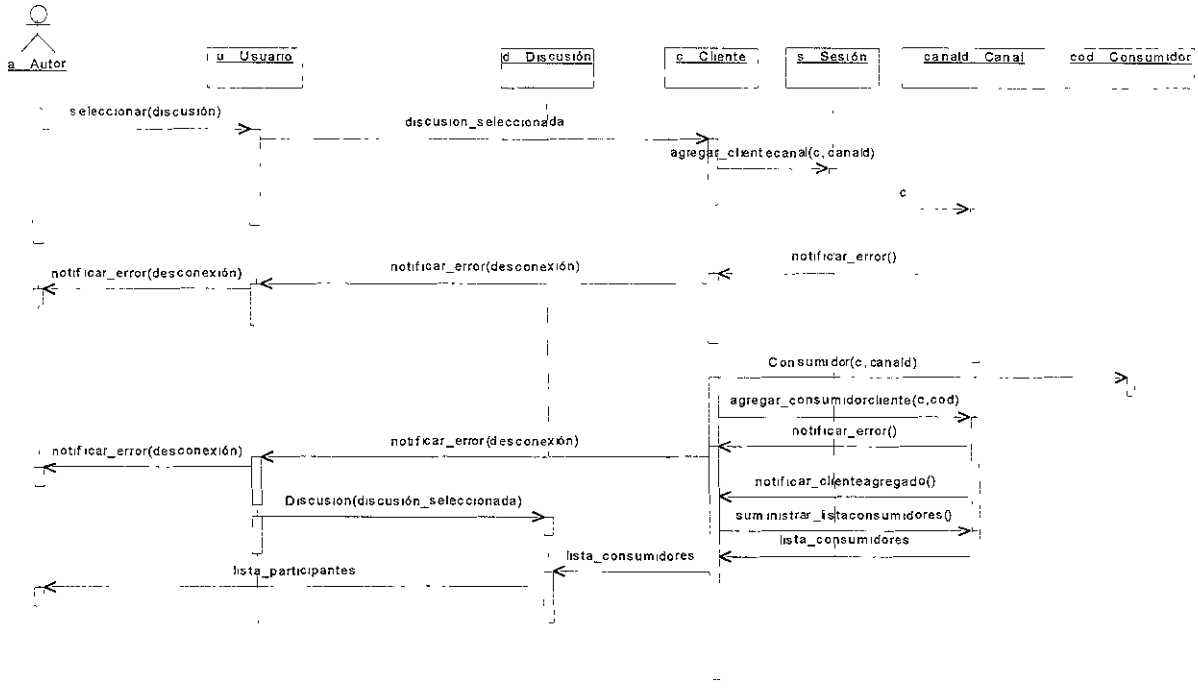


Figura 4.12. Diagrama de secuencia del escenario unirse a una discusión.

La figura 4.12 presenta el diagrama de secuencia del escenario unirse a una discusión. Nótese que la secuencia de este diagrama es muy similar a la secuencia que ocurre cuando un autor decide participar en una discusión generada (ver figura 4.11). La diferencia entre estas secuencias radica en que, en el escenario **unirse a una discusión**, el autor selecciona la discusión a la que se desea unir. Es importante resaltar que el autor se entera de las discusiones que se encuentran en proceso, en el momento en que realiza el escenario unirse a la sesión.

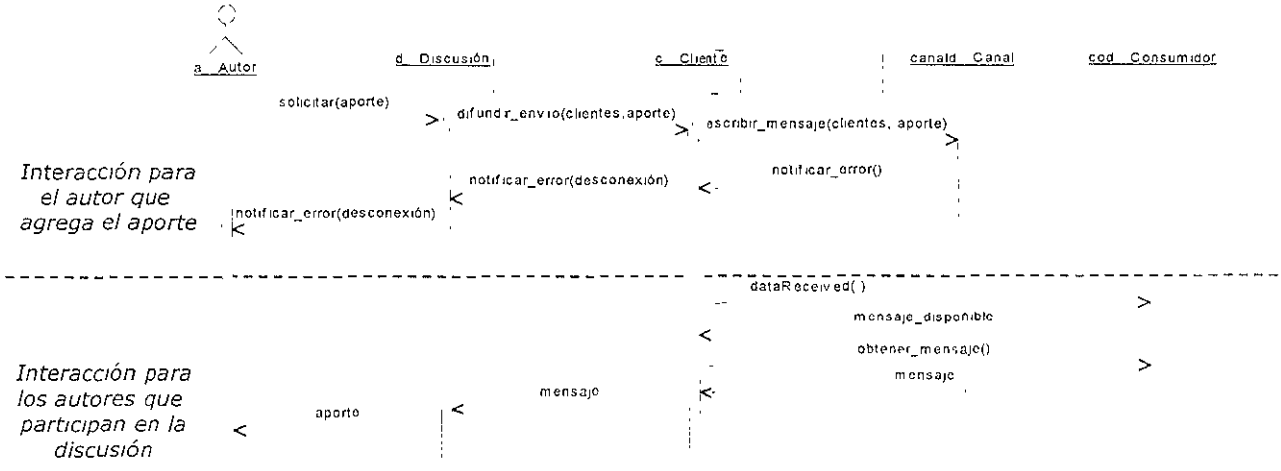


Figura 4.13. Diagrama de secuencia del escenario agregar aporte a la discusión.

La figura 4.13 presenta la secuencia que ocurre cuando los autores agregan sus aportes a la discusión. Nótese que la difusión de los aportes utiliza el mismo mecanismo de la difusión de mensajes entre los autores: los aportes son escritos como mensajes al canal de la discusión (canald), recibidos por los consumidores y desplegados en la interface gráfica de la discusión.

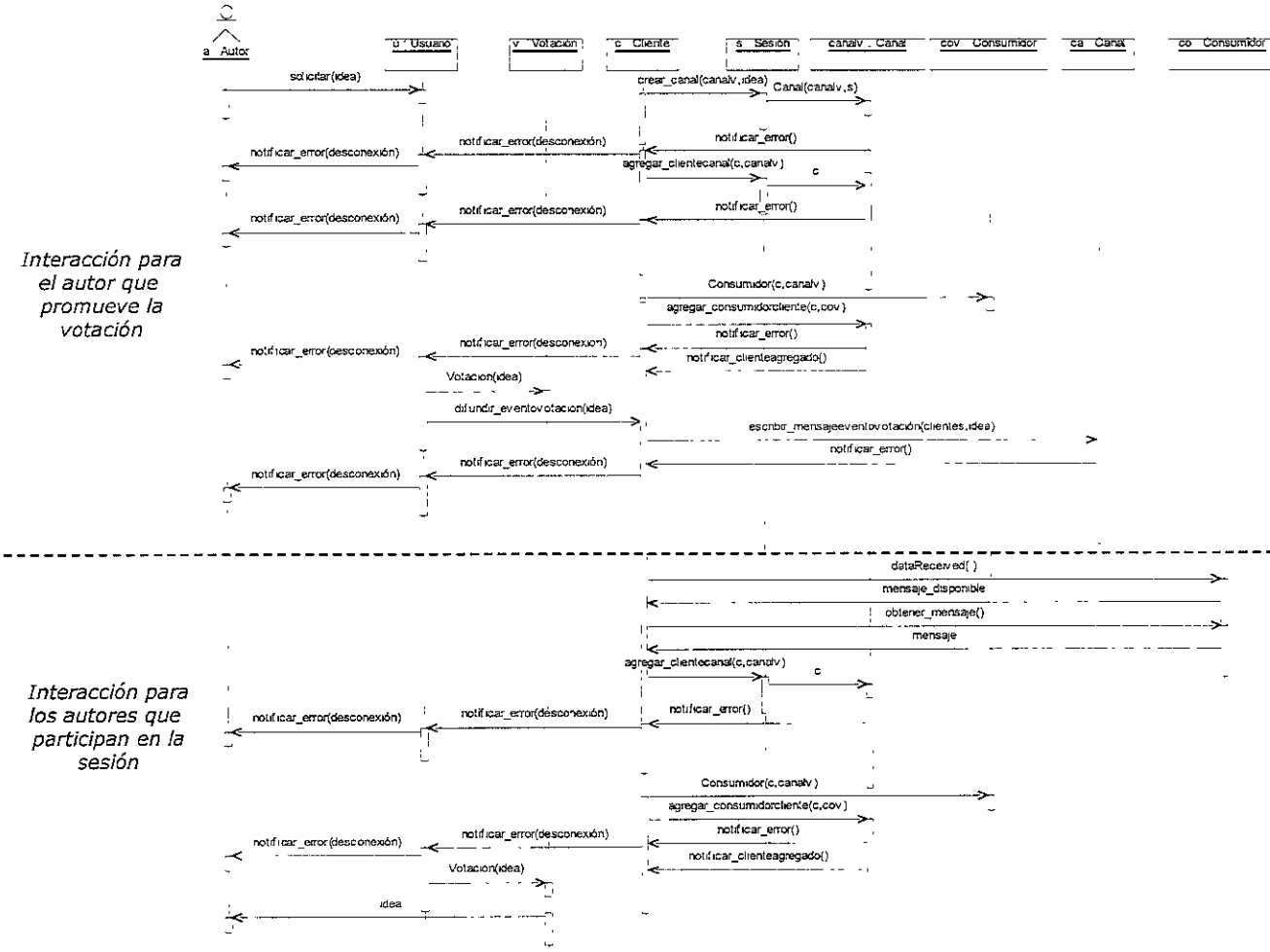


Figura 4.14. Diagrama de secuencia del escenario promover votación.

La figura 4.14 presenta la secuencia del escenario promover votación, la cual es similar a la presentada en el escenario **generar discusión**. Nótese que al igual que para generar una discusión, cuando un autor promueve una votación, se crea un objeto de la clase Canal, denominado **canalv** y otro de la clase Consumidor, denominado **cov**.

En este diagrama se destaca que para la recepción del evento de la votación promovida, los autores no realizan ninguna intervención, debido a que no se difunde una invitación a participar en la votación. Por ello, una vez que el objeto **c** (cliente) de cada autor recibe el mensaje del evento de votación, llama al método **agregar_clientecanal(c,canalv)** del objeto **s**, para ser agregado al canal de la votación.

Después de que cada cliente es agregado como consumidor del canal **canalv**, el objeto de la clase Usuario de cada autor, crea la interface gráfica de la votación (objeto de la clase Votación) y despliega la idea sometida a elección. Nótese que esta idea se difunde a cada cliente, mediante el mensaje de evento de la votación promovida.

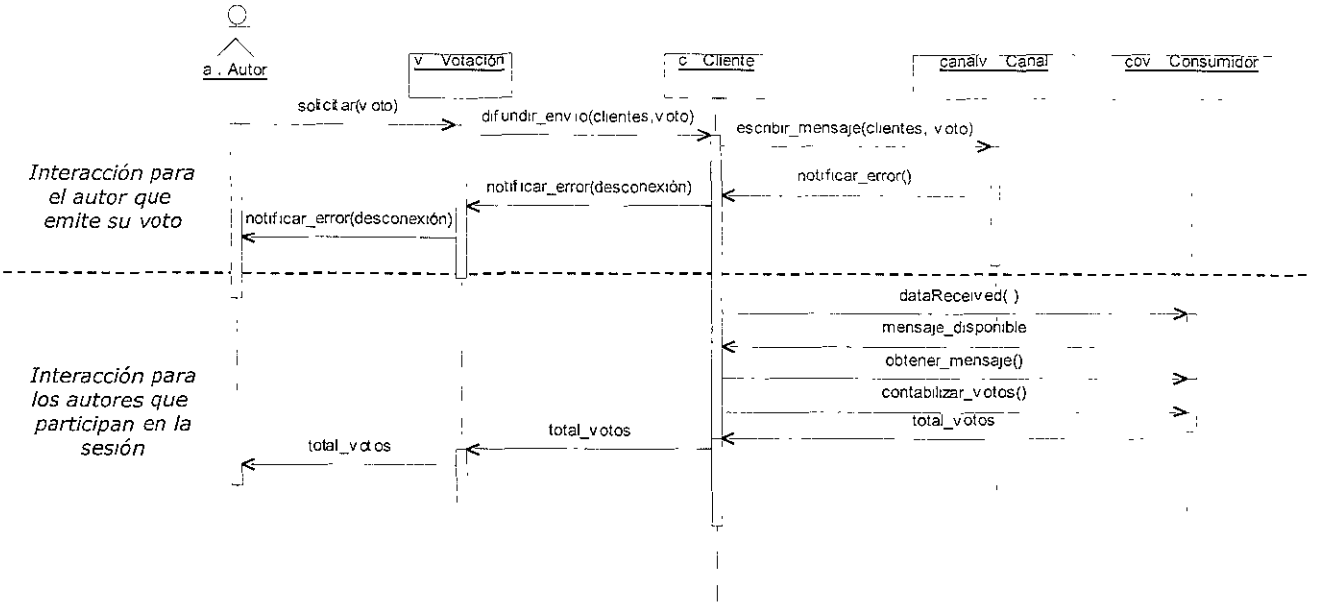


Figura 4.15. Diagrama de secuencia del escenario realizar votación.

En la figura 4.15 se presenta la secuencia del escenario realizar votación, en el cual cada autor emite su voto sobre la idea sometida a votación. Esta secuencia utiliza el mecanismo de difusión de mensajes para la difusión de los votos de cada autor.

En el diagrama se destaca que a diferencia de los escenarios anteriores, el mensaje no se despliega al autor tal y como se recibe. Cada cliente llama el método **contabilizar_votos()** del objeto **cov**, para que realice la suma del voto recibido y solamente se despliegue el resultado de la votación en la interface gráfica de la misma.

Con este diagrama, se finaliza la descripción de la interacción que existe entre todas las clases que conforman el espacio de trabajo compartido y los autores que se reúnen en la sesión de grupo.

Los diagramas obtenidos en esta etapa de análisis (casos de uso, clases y responsabilidades, y secuencias), establecen los aspectos para el diseño y la posterior construcción, de los mecanismos de colaboración y del espacio de trabajo compartido para Alliance.

4.3. Diseño

De acuerdo con la metodología unificada, la etapa de diseño involucra la elaboración de un modelo lógico y un modelo físico de las clases que componen un sistema. El modelo lógico especifica la organización de las clases, según su función en el dominio del problema y otros aspectos como la conexión a las bases de datos, el soporte para el ambiente gráfico, etc. El modelo físico especifica principalmente, la estructura de las clases y su organización en librerías o paquetes, para la construcción y la puesta en marcha del sistema.

En la presente investigación no se realiza la etapa de diseño exactamente como se sugiere en la metodología unificada, ya que el modelo lógico ha sido resumido en la sección 4.2.3 (clases y responsabilidades) y el modelo físico se resume en la sección 4.4.2, con la especificación de cada clase que compone el espacio de trabajo compartido.

De esta forma, la etapa de diseño se enfoca en la definición de la arquitectura que apoya el desarrollo del espacio de trabajo, los componentes de dicha arquitectura y las interfaces gráficas para los autores.

4.3.1. Arquitectura

La arquitectura híbrida del espacio de trabajo compartido, utiliza un modelo cliente-servidor que combina dos técnicas esenciales: el manejo centralizado de la sesión de grupo y la difusión de eventos en forma distribuida (ver sección 2.3.2).

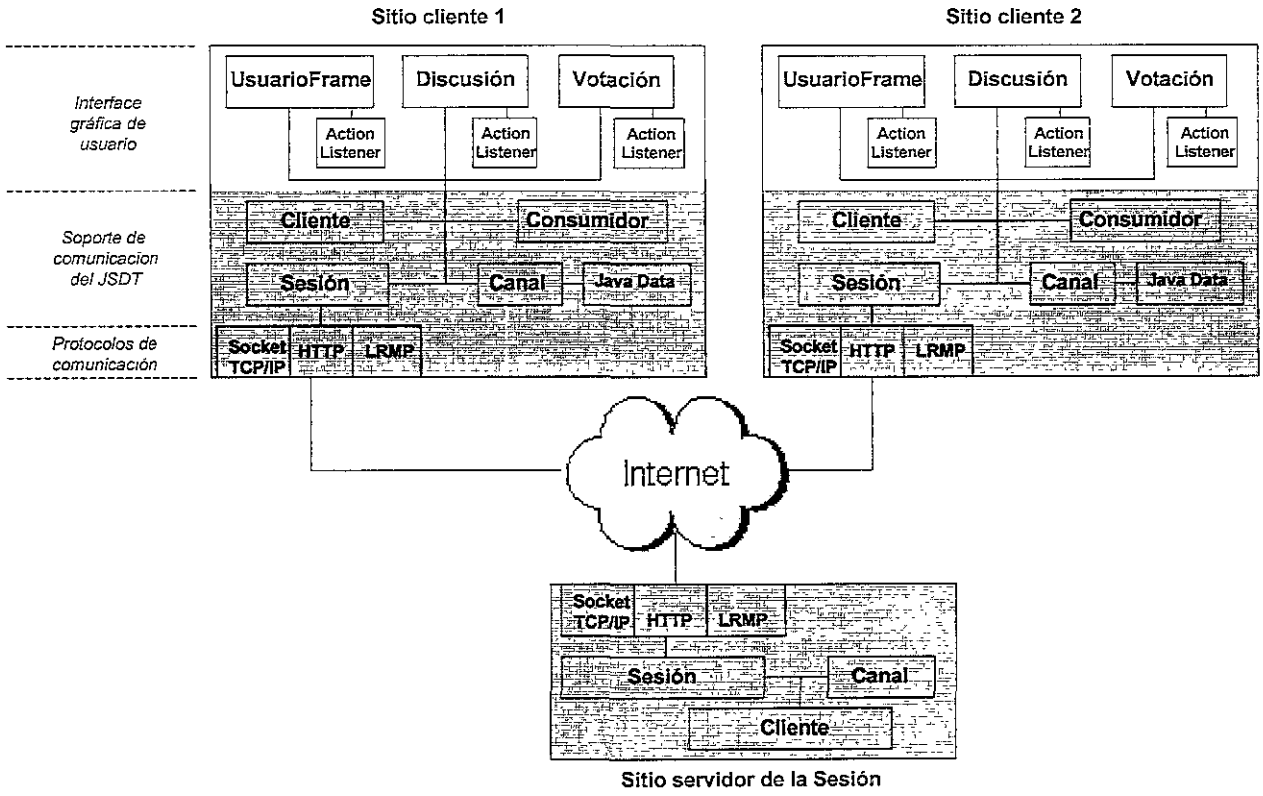


Figura 4.16. Arquitectura híbrida del espacio de trabajo compartido para Alliance.

En la figura 4.16 se presenta la arquitectura híbrida del espacio de trabajo compartido. La figura presenta dos sitios clientes (1 y 2) y un sitio servidor denominado **sitio servidor de la sesión.** El sitio servidor es el encargado de crear la sesión de grupo y los canales de comunicación para la difusión de los eventos de la sesión.

En cada sitio existe un conjunto de objetos de diversas clases, que desempeñan funciones específicas. Estos objetos están agrupados en dos tipos de funciones: la interface gráfica para

los autores y el soporte de comunicación del JSDT. Dicho soporte de comunicación incluye los protocolos utilizados para la sesión de grupo: *Socket*[2] TCP/IP, HTTP o LRMP[3].

4.3.1.1. Componentes

Los objetos componentes de la arquitectura del espacio de trabajo compartido se definen como sigue:

Sesión

El objeto sesión es fundamental para el establecimiento de la comunicación de grupos ya que constituye el punto de reunión de los clientes para que puedan interactuar entre sí. Este objeto tiene asociado un tipo especial de URL, el cual se define en el JSDT como:

jsdt://<servidor>:<puerto>/<implementación>/Session/<nombre>

en donde: <servidor> corresponde al nombre o la dirección IP de la computadora que funciona como servidor de la sesión JSDT, <puerto> corresponde al número del puerto TCP/IP utilizado para las conexiones al servidor, <implementación> corresponde al protocolo de comunicación utilizado y <nombre> corresponde al nombre de la sesión. Un ejemplo de un URL para especificar una sesión JSDT puede ser:

jsdt://mi.servidor.mx:4461/socket/Session/WSSesion.

El tipo de implementación de la sesión establece el protocolo que se utiliza para el intercambio de la información. Este tipo de implementación puede ser:

- **Sockets de TCP/IP**: para el establecimiento de canales confiables y no confiables, es decir, donde se garantiza o no la entrega del mensaje en el canal de comunicación. El uso de sockets mantiene abierta, en lo posible, la conexión entre el cliente y el servidor, para facilitar el envío del mensaje y la notificación de cada consumidor que lo recibe.
- **HTTP**: utilizando el método *HTTP-POST* para el establecimiento de canales confiables. El uso del HTTP no mantiene abierta la conexión entre el cliente y el servidor, sino que cada vez que se envía el mensaje y los consumidores notifican que lo han recibido, la conexión se cierra. Esto implica que por cada envío de cada mensaje, debe abrirse la conexión entre el cliente y el servidor, generando nuevos requerimientos *HTTP-POST*.
- **LRMP**: el cual es un protocolo para multicast sobre canales confiables. Este protocolo se basa en la difusión de mensajes en tiempo real e introduce un mecanismo de recuperación local de los mensajes, en caso de fallas en la red de comunicación. La desventaja en el uso de este protocolo, radica en que las prioridades asignadas a los objetos Data (se describe más adelante), que se transmiten por el canal, son ignoradas. Adicionalmente, el uso de LRMP implica que se definan en forma estática las direcciones multicast válidas para la difusión de mensajes, mientras que los sockets y el HTTP, manejan estas direcciones en forma directa de las referencias de los objetos registrados en la sesión.

El objeto sesión del sitio servidor define el nombre de la sesión y los parámetros establecidos para la comunicación, los cuales comprenden:

- El URL de la sesión (como se describió anteriormente),

[2] Un objeto de software que conecta un programa de aplicación con un protocolo de red, para el envío y recepción de datos a través de Internet. El socket provee las funciones necesarias para traducir los datos en mensajes TCP/IP

[3] Del término *"Light weight Reliable Multicast Protocol"*, http://webcanal.inria.fr/lrmp/index.html

- el objeto cliente que crea la sesión, por ejemplo cliente="servidor".

En cada sitio cliente, el objeto sesión se utiliza para unirse a la sesión de grupo y mantener una referencia del objeto sesión creado en el sitio servidor. Con ello, los objetos clientes pueden manipular el objeto sesión en forma local, es decir, pueden llamar sus métodos localmente. Sin embargo, estas referencias del objeto sesión presentan una dependencia total del objeto sesión del servidor, en operaciones que generan cambios en la sesión, por ejemplo cuando se crea un canal o se agrega un nuevo cliente, lo cual implica que el servidor debe ser contactado cada vez que se realiza dicho cambio.

Cliente

Es el objeto que identifica a cada autor que colabora en la sesión de grupo. Este objeto se utiliza para establecer el origen y el destino de los mensajes que se intercambian por medio del canal de comunicación.

Cada objeto cliente tiene asociado un nombre que debe ser único para la sesión de grupo. En el sitio servidor de la sesión, el objeto cliente es requerido exclusivamente para crear el objeto sesión y no participa en el intercambio de los mensajes. Debido a que cada sitio cliente posee su propio objeto cliente, no se requiere hacer referencia a otros objetos definidos en el servidor, como es el caso de los objetos sesión.

Canal

El objeto canal especifica una ruta de comunicación entre los clientes de una sesión. Para ello, el objeto sesión agrega los objetos cliente que pueden enviar mensajes a través de cada canal y mantiene un registro de ellos.

En el sitio servidor, el objeto canal establece un medio de comunicación para la difusión de los eventos de la sesión. En los sitios clientes, cada objeto canal puede establecer lo siguiente:

- Una ruta independiente sobre la cual se comparte la información de cada discusión generada o cada votación promovida.
- Una referencia al canal de la sesión creado por el servidor, para obtener la información relacionada con los eventos difundidos y la colaboración.
- Una referencia a otros objetos canal, creados por otros sitios clientes que han generado discusiones o promovido votaciones.

Cada canal creado en una sesión de trabajo es registrado en el servidor para mantener su referencia disponible a todos los sitios clientes. Por ello, la desconexión de un cliente que haya creado un canal para compartir información, no afecta la comunicación de los clientes que lo estén utilizando. El siguiente ejemplo explica esta situación:

- El sitio cliente 1 genera una discusión, esto es, crea un objeto discusión y un objeto canal que se agrega al registro de los objetos de la sesión en el servidor.
- El sitio cliente 2 se une a la discusión, lo cual significa que crea su objeto discusión y se agrega al canal creado para la discusión. Para ello, este sitio crea un objeto canal, el cual es una referencia del objeto canal creado en el sitio 1.
- Un nuevo sitio cliente 3 se une a la discusión, realizando el mismo proceso que se llevó a cabo para el sitio cliente 2.
- Después de un tiempo, el sitio cliente 1 se desconecta y los sitios cliente 2 y 3 continúan en la discusión, sin verse afectados por dicha desconexión.

- Los sitios cliente 2 y 3 continúan su interacción en la discusión, porque sus objetos canal mantienen la referencia (del canal creado) con base en el registro de objetos en la sesión y no con base en el objeto canal del sitio cliente 1.

Consumidor

Para que cada cliente pueda recibir la información que se envía a través de los canales, debe registrarse como entidad consumidora de los mismos, esto es, debe crear un objeto consumidor del correspondiente canal. Por esta razón, en el sitio servidor de la sesión no existe ningún objeto consumidor de ningún canal, puesto que el servidor solamente facilita el registro de los canales creados en la sesión, pero nunca consume ninguno de los mensajes que se envían a través de ellos.

El proceso de registrar cada cliente como consumidor de un canal específico es realizado por el objeto canal. Para realizar dicho registro, es necesario que en el sitio cliente existan los objetos cliente, consumidor y canal.

El objeto consumidor puede recibir diversos tipos de mensajes de uno o de varios canales. Sin embargo, como se verá en los aspectos de la construcción (sección 4.4.2), resulta ser más eficiente utilizar un tipo de consumidor para cada canal, ya que se reduce la complejidad de las operaciones que debe llevar a cabo cada consumidor.

Java Data

Un objeto Data del lenguaje Java es un vector de bytes mediante el cual se codifican los mensajes que se transmiten por un canal, a uno o a varios clientes consumidores. Este objeto tiene los siguientes atributos:

- Un arreglo de bytes, que corresponde al mensaje.
- La longitud del arreglo de bytes.
- La prioridad que se asigna al objeto Data que se transmite por medio del canal. Esta prioridad puede ser: TOP_PRIORITY, HIGH_PRIORITY, MEDIUM_PRIORITY y LOW_PRIORITY, indicando un valor de prioridad, desde el más alto al más bajo respectivamente. La prioridad del objeto Data es muy útil cuando se envía un gran número de mensajes sobre un canal y se requiere asignar un orden específico para su recepción.
- El nombre del cliente que envía el objeto Data.
- El canal sobre el cual se envían el objeto Data.

De esta forma, cuando un consumidor recibe un mensaje de un canal, no sólo recibe el mensaje enviado sino que además puede determinar información adicional, como por ejemplo el cliente emisor.

Objetos de las interfaces gráficas para el autor

Los objetos usuario, discusión y votación, realizan el despliegue de las interfaces gráficas de la ventana principal, la ventana de discusión y la ventana de votación, respectivamente. Estos objetos tienen en común un componente denominado controlador de acciones o *ActionListener*[4]. Este componente se utiliza implementando el método **ActionPerformed()** en cada objeto de interface gráfica, con el cual se capturan los eventos que generan los elementos que la conforman (botones, menús, listas, etc.).

[4] Interface definida para la clase java.awt event

Cada elemento de la interface gráfica se agrega al ActionListener para que sea detectado por el método ActionPerformed(). Por ejemplo, cuando el autor presiona un botón o selecciona una opción del menú, el método ActionPerformed() verifica el elemento que realizó la acción y ejecuta las operaciones que se definan para ella.

El ActionListener detecta exclusivamente los eventos que ocurren en forma local en cada interface gráfica del sitio cliente. Por ello, para difundir un evento ocurrido, como por ejemplo realizar la votación, se debe combinar el uso del ActionListener (procesar el evento de manera local), el envío del mensaje relativo al evento y el método dataReceived() de los consumidores que reciben el mensaje.

Los objetos de la interface gráfica constituyen los componentes del nivel más alto de la arquitectura del espacio de trabajo compartido. Dichos objetos contienen el conjunto de elementos que se agregan a la interface de ventanas, para generar el espacio de interacción para los autores.

4.3.1.2. Manejo centralizado de la sesión

En la arquitectura del espacio de trabajo (ver figura 4.16), el sitio servidor de la sesión no presenta objetos relacionados con la interface gráfica para los autores, sino que solamente contiene objetos para el soporte de comunicación del JSDT. Esto se debe a que el servidor es un proceso centralizado, el cual se encarga de almacenar y mantener disponible a los sitios clientes, toda la información de los objetos relacionados con la sesión, como lo son: clientes conectados, canales creados, consumidores, etc.

A pesar de que el servidor es un proceso centralizado, este proceso puede ejecutarse en cualquiera de los sitios clientes. Sin embargo, dicho proceso sólo se ejecuta en un sitio cliente a la vez. Esto quiere decir que los sitios clientes no llegan a ser sitios cliente-servidor, sino que debe existir solamente un proceso servidor de la sesión, que puede ejecutarse en cualquier cliente, para que los demás sitios clientes puedan interactuar entre sí. En la figura 4.16 se encuentra separado el sitio servidor de la sesión de los sitios clientes, con el fin de que se ejecuta en forma centralizada.

4.3.1.3. Difusión distribuida de eventos

La difusión de los eventos se realiza en forma distribuida desde cada sitio cliente. Para ello, se utilizan los controladores de acciones ActionListener de cada interface gráfica, los cuales detectan cada evento que ocurre dentro de ellas (botón presionado, opción de menú seleccionada, etc.). De acuerdo al evento detectado, el controlador de acciones genera la escritura del mensaje de dicho evento en el canal de comunicación. Una vez se escribe el mensaje en el canal cada sitio cliente utiliza su objeto consumidor para obtener el mensaje, procesarlo y desplegarlo en la respectiva interface gráfica.

Para ejemplificar la difusión de eventos en la arquitectura del espacio de trabajo compartido para Alliance, considere el proceso de agregar un aporte a una discusión:

- En la interface gráfica de la discusión (objeto de la clase Discusión) del sitio cliente 1, se detecta el evento de envío de un aporte, mediante su correspondiente controlador de acciones.
- El controlador de acciones genera la orden de envío del mensaje relacionado con el evento de agregar aporte, a través del canal creado para la discusión.

- El mensaje se codifica utilizando el objeto Java Data, para ser transmitido hacia el servidor de la sesión. La transmisión se hace hacia el servidor de la sesión, porque éste es quien mantiene la referencia principal del canal de la discusión y sus consumidores.
- Para la escritura del mensaje en el canal, el sitio cliente 1 utiliza el objeto cliente y el objeto representante de la sesión, el cual tiene asociado el protocolo de comunicación utilizado.
- Una vez escrito el mensaje en el canal, el servidor de la sesión notifica al consumidor del sitio cliente 2 la escritura de dicho mensaje. Para ello, el servidor utiliza el registro de los clientes que son consumidores del canal de la discusión.
- Cuando el consumidor del sitio cliente 2 recibe la señal de que se escribió el mensaje, envía al servidor una señal de recibido y ejecuta la especificación de su método dataRecieved(). Con ello, el consumidor obtiene el mensaje escrito en el canal, lo decodifica y genera un evento de aporte agregado en la interface gráfica de la discusión.

La difusión de los eventos se realiza en forma distribuida porque cada sitio cliente realiza la escritura del mensaje en el correspondiente canal. El servidor de la sesión solamente facilita la entrega de dicho mensaje a los consumidores de los sitios clientes, para que procesen en forma local los eventos recibidos. En ninguno de los casos el servidor genera un evento directo hacia los sitios clientes (por ejemplo actualizar una interface gráfica), sino que funciona como intermediario en la difusión de los mensajes, suministrando las referencias a los objetos de la sesión.

Adicionalmente, los objetos discusión y votación de cada sitio cliente, crean en forma dinámica sus propios canales de comunicación para compartir la información. Por ello, la creación de dichos canales se realiza en forma distribuida e implica que cada sitio cliente difunda un evento para que los demás sitios se agreguen al canal creado. Sin embargo, cada canal creado debe ser registrado en el servidor de la sesión para mantener su referencia disponible a todos los sitios clientes.

Durante la difusión de los eventos se presentan dos formas de interacción, entre los objetos que componen los sitios clientes:

- La difusión y la percepción de los eventos, a nivel de los objetos cliente, se llevan a cabo en forma síncrona débil y fuertemente acoplada: cuando un objeto cliente difunde un evento, esta difusión se realiza en forma síncrona débilmente acoplada, porque depende de un evento generado por una acción voluntaria del autor. Cuando el evento es recibido y procesado en cada cliente destino, la percepción del evento se realiza en forma síncrona fuertemente acoplada, porque no requiere de una acción voluntaria de recepción.

- La transmisión de los mensajes se lleva a cabo de manera asíncrona: el consumidor recibe la señal (enviada por el servidor) de que hay un mensaje disponible, después de que dicho mensaje ha sido escrito en el canal. El cliente que escribe el mensaje espera el período de TimeOut para recibir las señales de cada consumidor que lo recibe (usando el servidor como intermediario). En ninguno de los casos existe una negociación previa, entre el cliente emisor y cada consumidor, para enviar y recibir de manera síncrona cada byte del mensaje que se transmite.

4.3.1.4. Manejo de la consistencia

En la sección 2.1.2 se discutió que la interacción de los usuarios dentro de un espacio de trabajo compartido requiere la coordinación de los eventos que se generan en forma simultánea, para resolver las inconsistencias en la información que se comparte o en el estado del espacio de trabajo. Esta coordinación de eventos se realiza mediante la adopción de

mecanismos para el control de la concurrencia [GrMa94], como por ejemplo bloqueos, transacciones atómicas o serialización, entre otros.

En la arquitectura híbrida definida la difusión distribuida de eventos se basa concretamente en el intercambio de mensajes sobre canales confiables, adoptando una política optimista para garantizar la escritura y la recepción de cada mensaje.

Con base en ello y dado que el manejo de la sesión se realiza en forma centralizada, el problema de la consistencia en esta arquitectura, se reduce al control de la concurrencia requerido para las operaciones de escritura y recepción de mensajes sobre los canales comunes.

El soporte de comunicación del JSDT incorpora un mecanismo para controlar dicha concurrencia, el cual utiliza un esquema de ordenamiento total de los mensajes que se intercambian en los canales comunes. Este mecanismo cumple las siguientes propiedades:

- **Serialización**: los mensajes concurrentes correspondientes a los eventos generados son escritos en forma secuencial en el canal común y recibidos en forma secuencial por cada consumidor del mismo. Con ello, el canal garantiza que los mensajes sean entregados en el mismo orden en que fueron escritos. Asimismo, el método dataReceived() del componente Consumidor, es un método que se define como **sincronizado**, con lo cual se *garantiza que cada llamada se ejecute completamente antes de ejecutar la siguiente* llamada.

- **Aislamiento**: cada mensaje escrito en el canal es manipulado como un mensaje independiente y el establecimiento del orden total permite que cada uno sea procesado por separado.

- **Atomicidad**: la escritura y recepción del mensaje es un proceso atómico donde se garantiza la entrega del mensaje a todos los consumidores activos del canal o no se entrega a ninguno. Si eventualmente un consumidor se desconecta de la sesión después del tiempo de TimeOut, ya no es considerado como consumidor activo del canal. Para implementar esta propiedad de atomicidad, las referencias del objeto sesión permiten identificar el estado de un canal y de los clientes que lo consumen.

Es importante resaltar que este mecanismo para el control de la concurrencia es suficiente para garantizar la consistencia en la difusión de eventos en forma distribuida, considerando que los sitios clientes y servidor mantiene un enlace de comunicación estable y la tolerancia a fallas no es un asunto crítico.

4.3.2. Interfaces de usuario del espacio de trabajo compartido

El espacio de trabajo compartido se basa en el concepto de sesión de grupo, para representar el lugar donde los autores se reúnen y llevan a cabo su interacción. En este espacio, se proporciona la información compartida mediante tres ventanas de tipo WYSIWIS, donde cada autor visualiza la misma información. Dichas ventanas comprenden: la ventana principal del espacio de trabajo, la herramienta de discusión y la herramienta de votación.

Adicionalmente, el espacio de trabajo proporciona una herramienta para el intercambio de mensajes síncronos entre los autores, la cual utiliza dos ventanas de diálogo, para solicitar el mensaje y para su despliegue. Dichas ventanas de diálogo también apoyan los procesos de generar discusiones y promover votaciones.

Las herramientas del espacio de trabajo se utilizan en forma colaborativa, esto es, cada autor las agrega al espacio en el momento que las requiere, promoviendo su interacción con los demás autores a través de ellas.

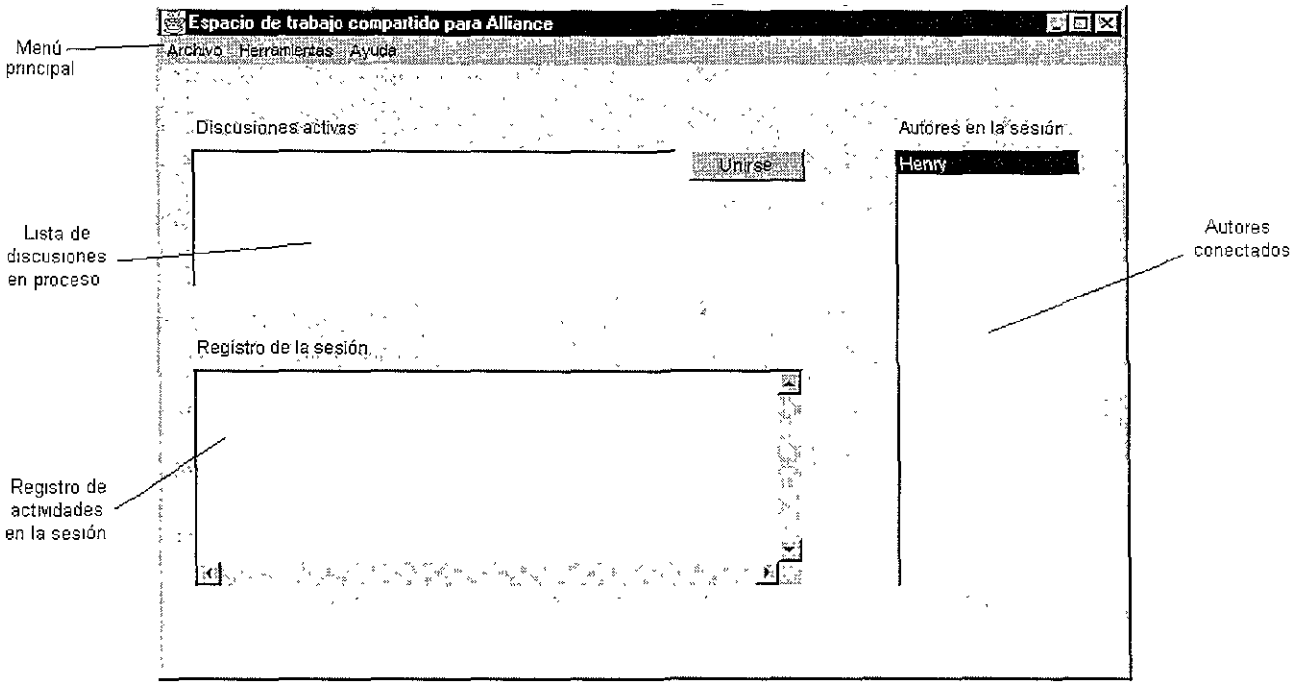


Figura 4.17. Interface gráfica de la ventana principal del espacio de trabajo compartido.

En la figura 4.17 se presenta la ventana principal de la interface gráfica del espacio de trabajo compartido para Alliance. El menú principal presenta las opciones: **Archivo**, para que el autor se conecte o se desconecte de la sesión de grupo, **Herramientas**, para que el autor genere discusiones, promueva votaciones o envíe mensajes, y **Ayuda**, para que el autor conozca el funcionamiento del espacio de trabajo.

Esta ventana incorpora tres herramientas simples para promover la conciencia de grupo síncrona: la lista de las discusiones que se llevan a cabo (activas), la lista de los autores en el espacio (en la sesión) y el registro de actividades en la sesión, el cual incluye los eventos de generar discusiones y promover votaciones, con el nombre del autor que los produce.

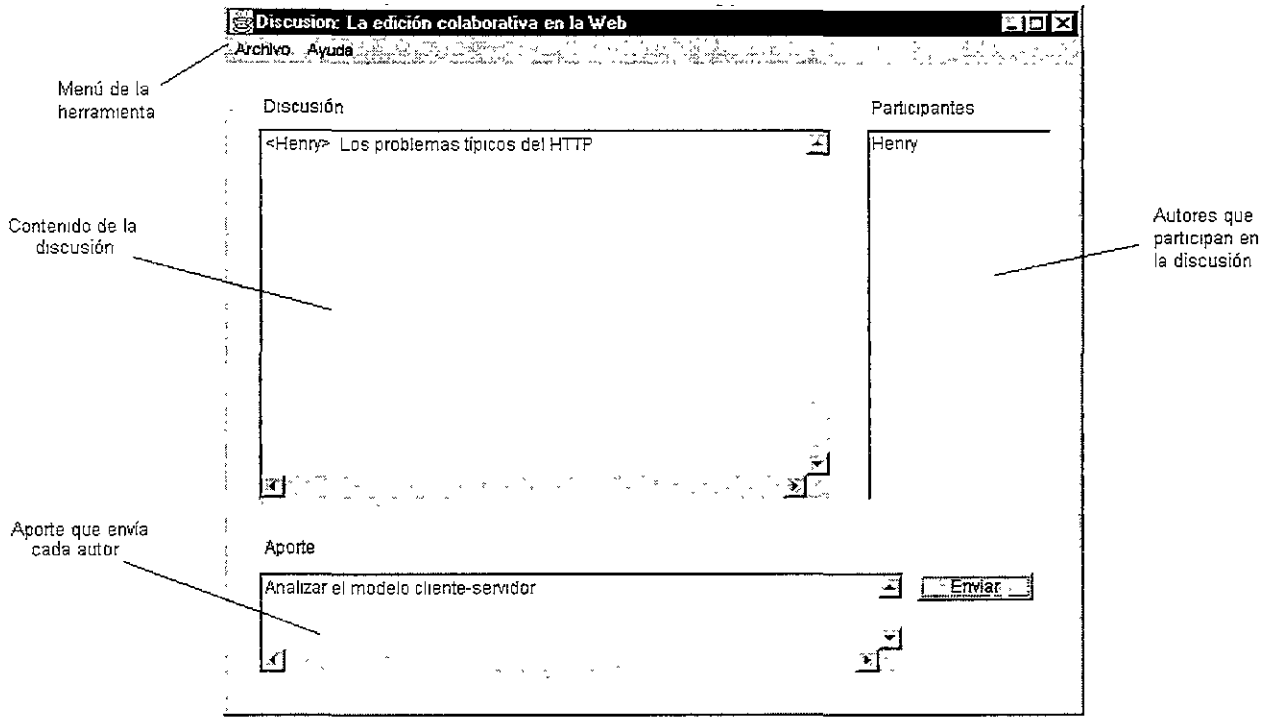


Figura 4.18. Interface gráfica de la ventana de cada discusión.

La figura 4.18 presenta la ventana que se despliega para cada discusión. El menú de la herramienta presenta las opciones: **Archivo**, para que el autor almacene los aportes de las discusiones, y **Ayuda**, para que el autor conozca el funcionamiento de la herramienta.

Para promover la conciencia de grupo síncrona, la ventana de la discusión incorpora la lista de los autores que participan en la discusión. Asimismo, el contenido de la discusión se presenta como un registro, el cual es difundido en forma síncrona a todos los autores participantes.

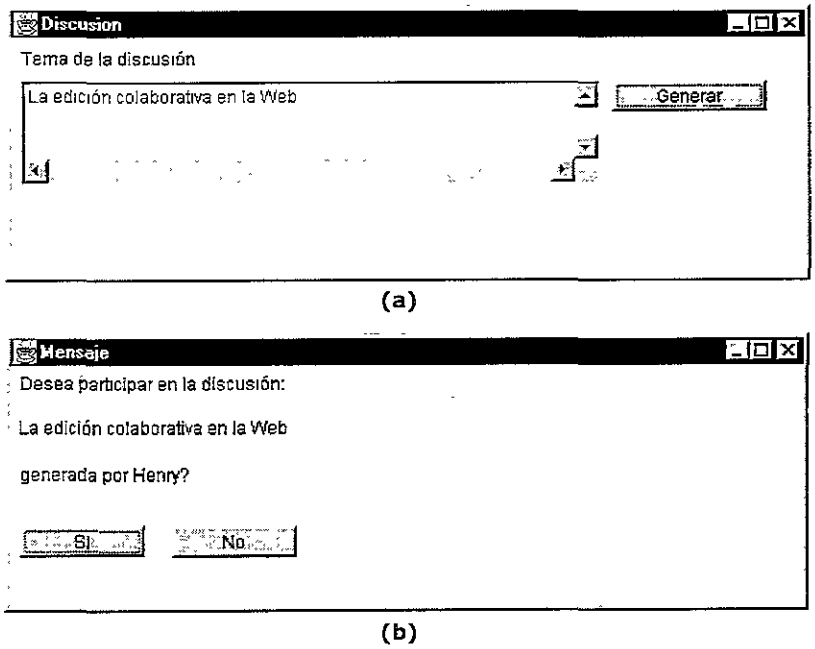


Figura 4.19. Ventanas de diálogo que apoyan el proceso de generar una discusión.

En la figura 4.19 se presentan dos ventanas de diálogo que se utilizan para apoyar el proceso de generar una discusión. En la parte (a) de la figura, se presenta la ventana que solicita el tema al autor que genera la discusión. En la parte (b) se presenta el mensaje que se despliega en forma síncrona, a los demás autores en la sesión, para que decidan si participan o no en la discusión generada.

La ventana de diálogo de la parte (b) de la figura, es desplegada como ventana activa para cada autor, independientemente de la actividad que esté realizando. Con ello, cada autor se entera del evento de discusión generada por uno de sus colegas.

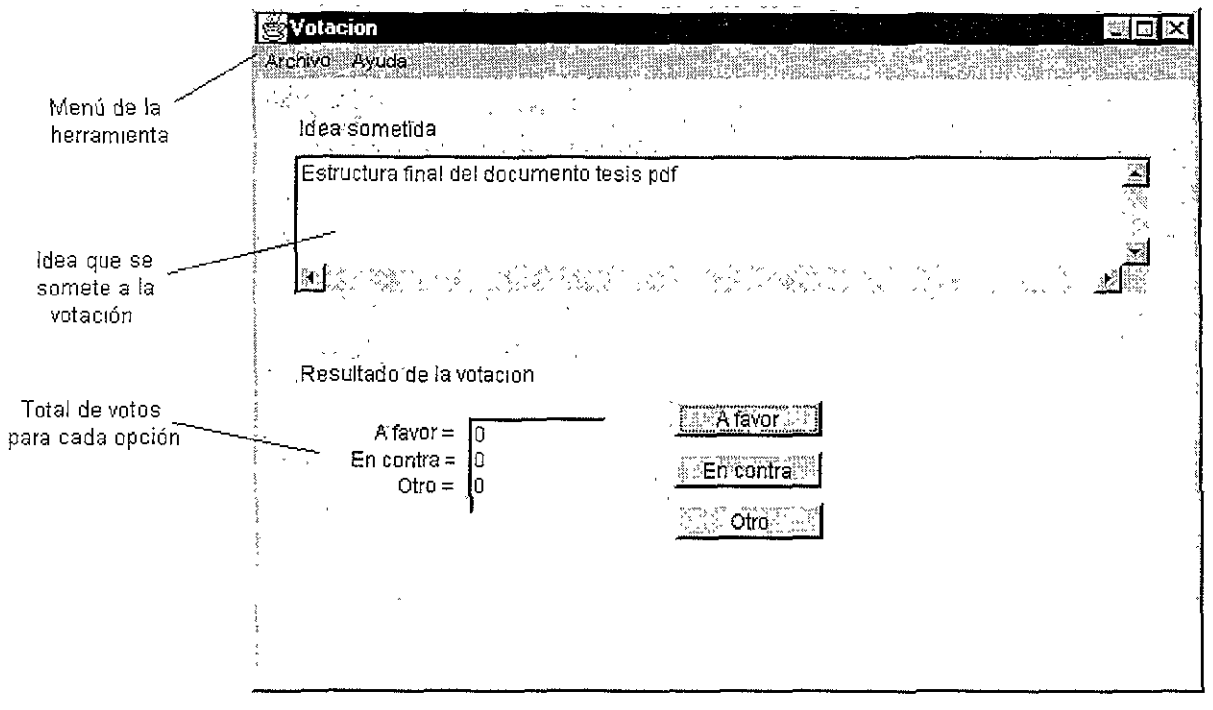


Figura 4.20. Interface gráfica de la ventana de cada votación.

La figura 4.20 presenta la ventana que se despliega a todos los autores en la sesión, cada vez que se promueve una votación. El menú de la herramienta incluye las opciones: **Archivo**, para que el autor almacene el resultado de la votación, y **Ayuda**, para que el autor conozca el funcionamiento de la herramienta. Además, esta ventana incluye la idea sometida a votación y el resultado de la misma. Dicho resultado se despliega en una lista que se actualiza en forma síncrona cuando cada autor emite su voto.

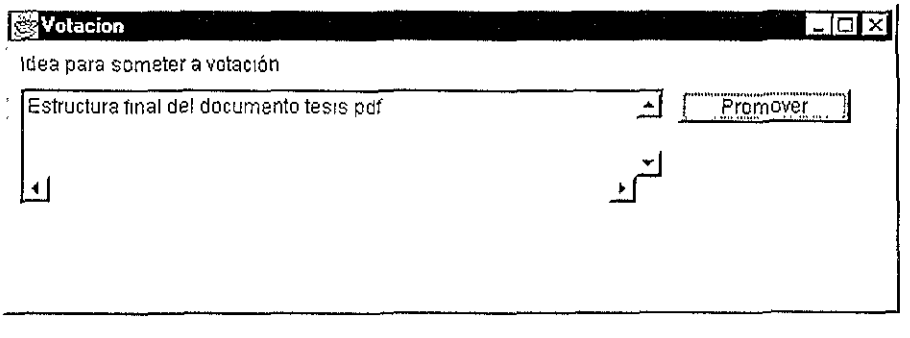


Figura 4.21. Ventana de diálogo que apoya el proceso de promover una votación.

Al igual que al generar una discusión, existe una ventana de diálogo que apoya el proceso de promover la votación, la cual se presenta en la figura 4.21. Esta ventana se encarga de solicitar al autor, la idea para promover la votación.

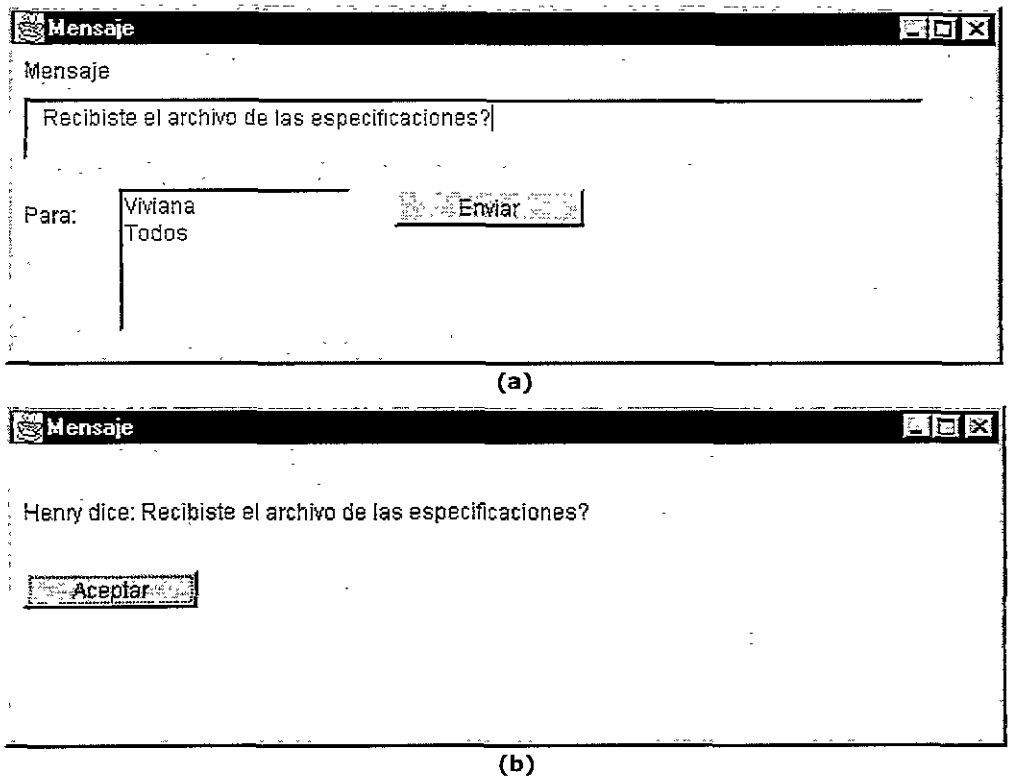


Figura 4.22. Ventanas de diálogo de la herramienta de mensajes.

Finalmente, en la figura 4.22 se presentan las ventanas de diálogo de la herramienta de mensajes. La parte (a) de la figura presenta la ventana que solicita el mensaje y su destinatario, y la parte (b) presenta la ventana que despliega el mensaje en forma síncrona.

4.4. Construcción

La etapa de construcción constituye la etapa final del desarrollo del espacio de trabajo compartido. Esta etapa se enfoca en dos aspectos principales: establecer el ambiente computacional para la programación y compilación de las clases en el lenguaje Java y especificar la organización física de los archivos relativos a cada clase (extensión .class en Java).

En la presente investigación se ha optado por incluir en el Apéndice 1, las especificaciones de implementación de cada clase, y en el Apéndice 2, las consideraciones para el uso del API JSDT. El código objeto generado en este desarrollo se encuentra disponible en el URL:

http://beynac.iimas.unam.mx/~lascaux/students/henry/tesis.html.

A continuación se presentan los aspectos para la construcción del espacio de trabajo compartido para Alliance.

4.4.1. Ambiente computacional

Para la construcción de las clases que conforman el espacio de trabajo compartido, se tienen los siguientes requerimientos de hardware y software:

- Hardware: una computadora personal, con procesador Pentium a 266 MHz o superior, 32 MB en memoria RAM, al menos 300 MB libres en disco duro y una tarjeta de red de 10 Mbs o superior.
- Software: sistema operativo Windows 9.X/NTWS, soporte de TCP/IP, dirección IP, herramienta de desarrollo Java Developers Kit (JDK) versión 1.2.1 o superior y el API Java Shared Data Toolkit (JSDT) versión 2.0 o superior. Tanto el JDK como el JSDT pueden obtenerse de manera gratuita del URL **http://java.sun.com/products**.

4.4.2. Especificación de las clases

En el espacio de trabajo compartido se utilizan diferentes canales para compartir la información de la sesión, y canales independientes para cada discusión y cada votación. A su vez, se definen varios tipos de consumidores que reciben los mensajes de un canal determinado. Esto simplifica las operaciones del método dataReceived() porque cada consumidor recibe sólo los mensajes referentes a determinados eventos. De esta forma se definen los siguientes canales para la sesión:

- **WSCanal**: se utiliza para compartir la información relacionada con el registro de las actividades que se llevan a cabo en la sesión.
- **EVCanal**: se usa para difundir los eventos relacionados con: el envío de mensajes entre los autores, las invitaciones a participar en las discusiones y las votaciones promovidas.
- **LUCanal**: se utiliza para compartir la lista de los autores que se encuentran en la sesión.
- **LDCanal**: se utiliza para compartir la lista de las discusiones que se encuentran en proceso (activas).

Para estos canales, se definen las clases: **WSConsumidor**, **EVConsumidor**, **LUConsumidor** y **LDConsumidor**, respectivamente. Cada cliente que se une a la sesión, se agrega a los canales de la misma y crea un objeto de cada clase consumidor, para ser agregado a cada canal.
Cada discusión crea en forma dinámica dos tipos de canales:

- Para compartir los aportes de cada discusión. El nombre de este canal se crea con el **tema** de cada discusión y el caracter "**D**" como prefijo, para distinguirlo de los demás canales de la sesión.
- Para compartir la lista de los autores que participan en cada discusión. El nombre de este canal se crea con el **tema** de la discusión y el prefijo "**LUD**", para diferenciarlo de los demás canales de la sesión.

Para estos canales, se definen las clases: **DiscConsumidor** y **LUDConsumidor**, respectivamente. Cada cliente que genera o que decide participar en una discusión, se agrega a los canales de la misma y crea un objeto de cada clase consumidor.

Cada votación promovida, crea en forma dinámica, un canal para compartir el resultado de la misma. El nombre de este canal se crea con la **idea** sometida a votación y el caracter "**V**" como prefijo, para diferenciarlo de los demás canales de la sesión. A su vez, cada cliente agregado al canal de cada votación, crea un objeto de la clase **VotConsumidor**.

De esta forma, se definen las siguientes clases para la construcción del espacio de trabajo:

- **WSCliente**: implementa la interface Client del JSDT, define el atributo **nombre**, que será asignado a cada cliente y el método getName() para regresar dicho nombre a los objetos que lo necesiten.
- **WSServidor**: crea la sesión WSSesion y los canales WSCanal, EVCanal, LUCanal y LDCanal. Verifica que el Registry está iniciado o de lo contrario lo inicia.
- **WSConsumidor**: implementa la interface ChannelConsumer para el registro de actividades en la sesión. Recibe los mensajes enviados por el canal WSCanal.
- **EVConsumidor**: implementa la interface ChannelConsumer para obtener los eventos difundidos en la sesión, los cuales comprenden votación promovida, invitación a discusión y mensajes entre los autores. Recibe los mensajes enviados por el canal EVCanal.
- **LDConsumidor**: implementa la interface ChannelConsumer para obtener la lista de discusiones en proceso. Recibe los mensajes enviados por el canal LDCanal.
- **LUConsumidor**: implementa la interface ChannelConsumer para obtener la lista de autores en la sesión. Recibe los mensajes enviados por el canal LUCanal.
- **WSUsuario**: realiza el proceso general de conexión y desconexión a la sesión y a los canales WSCanal, LUCanal, LDCanal y EVCanal.
- **Discusion**: realiza el proceso general de la discusión y construye la respectiva ventana.
- **DiscConsumidor**: implementa la interface ChannelConsumer para obtener los aportes enviados a cada discusión. Recibe los mensajes enviados por el canal de la discusión.
- ***LUDConsumidor***: *implementa la interface ChannelConsumer para obtener la lista de* autores en la discusión. Recibe los mensajes enviados por el canal creado para dicha lista de participantes.
- **Votacion**: realiza el proceso general de la votación y construye la respectiva ventana.
- **VotConsumidor**: implementa la interface ChannelConsumer para obtener el resultado de cada votación. Recibe los mensajes enviados por el canal de la votación.
- **EVDialogo**: construye la ventana (Frame) para desplegar los mensajes difundidos a uno o a todos los autores en la sesión.
- **MSGDialogo**: construye la ventana (Frame) que solicita el mensaje para ser enviado a uno o a todos los demás autores.
- **WSDialogo**: construye la ventana (Frame) que solicita el tema para generar la discusión o la idea para promover la votación.
- **WSUsuarioFrame**: construye la ventana (Frame) principal del espacio de trabajo compartido.

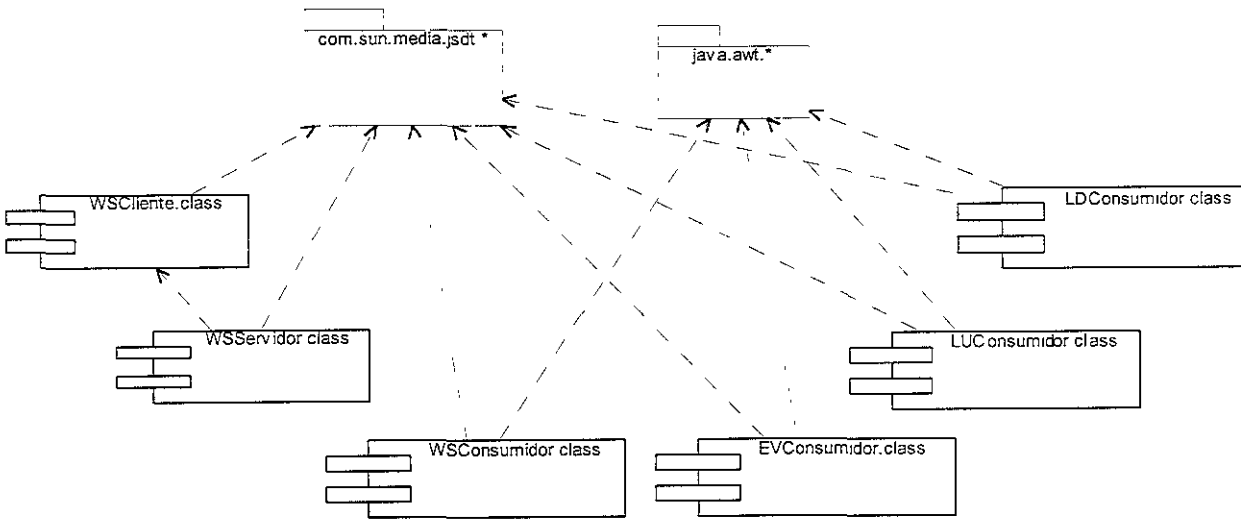


Figura 4.23. Diagrama de los componentes WSServidor, WSCliente, WSConsumidor, EVConsumidor, LDConsumidor y LUConsumidor.

La figura 4.23 presenta las dependencias de las clases WSServidor, WSCliente, los cuatro consumidores básicos de los canales de la sesión y los paquetes **java.awt** y **com.sun.media.jsdt**. Con estas dependencias se presenta la forma como debe compilarse cada clase componente. Por ejemplo, para poder compilar la clase WSServidor, debe compilarse primero la clase WSCliente.

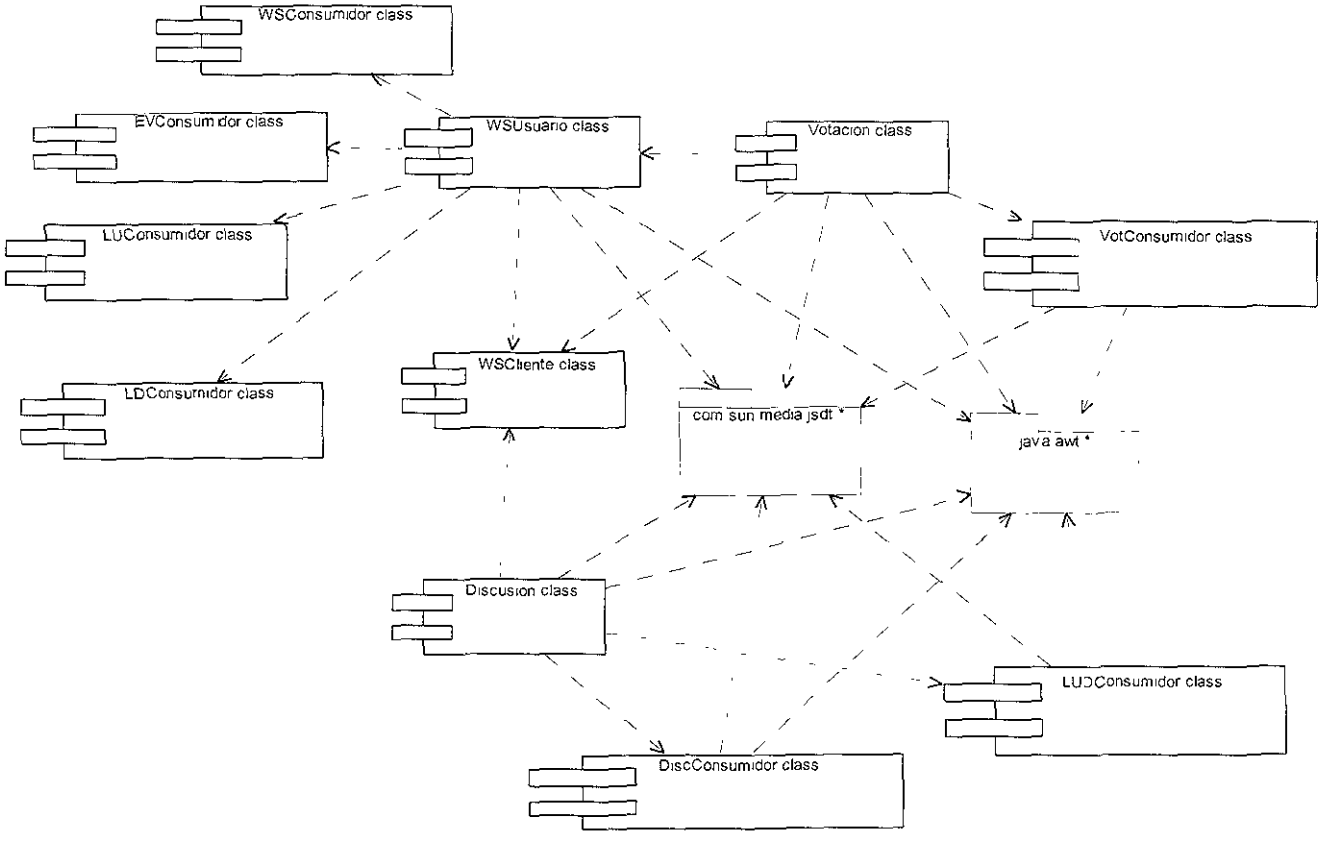


Figura 4.24. Diagrama de los componentes WSUsuario, Discusión, DiscConsumidor, LUDConsumidor, Votación y VotConsumidor.

La figura 4.24 presenta las dependencias de las clases WSUsuario, Discusión, Votación, DiscConsumidor, LUDConsumidor y VotConsumidor. Nótese que para la compilación de la clase WSUsuario, se requiere que previamente se compilen los consumidores básicos de los canales de la sesión y la clase WSCliente. De igual forma, las clases Votación y Discusión, requieren la compilación previa de los consumidores para los canales que utilizan.

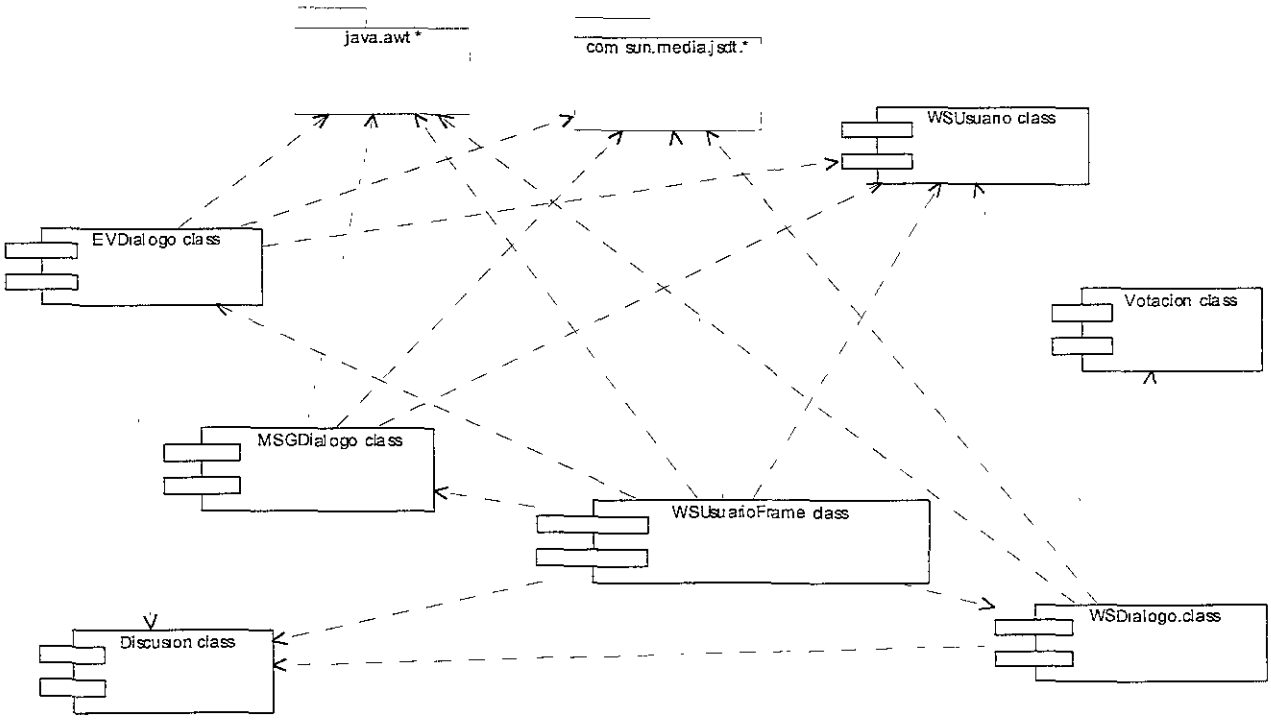


Figura 4.25. Diagrama de los componentes WSDialogo, EVDialogo, MSGDialogo y WSUsuarioFrame.

Finalmente, la figura 4.25 presenta las dependencias de las clases WSDialogo, EVDialogo, MSGDialogo y WSUsuarioFrame. Nótese que estos componentes corresponden a la ventana principal del espacio de trabajo compartido, y que la clase WSUsuarioFrame es la última en el orden de la complicación.

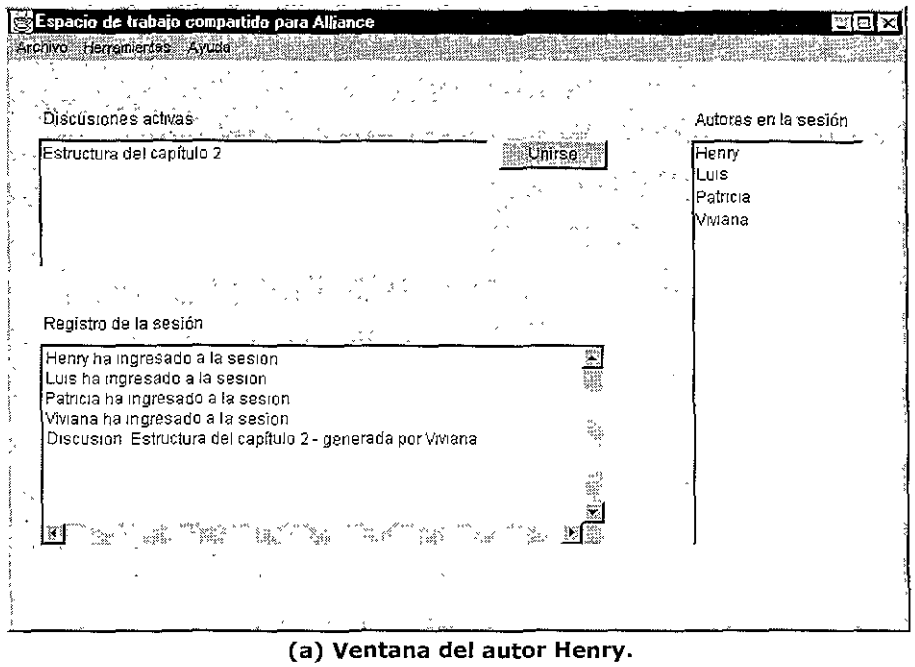
4.5. Evaluación del espacio de trabajo compartido

Las características del espacio de trabajo compartido para Alliance se basan en el marco teórico presentado en los capítulos 2 y 3, enfatizando los aspectos de interacción y de conciencia de grupo, involucrados en la fase de negociación entre un grupo de autores.

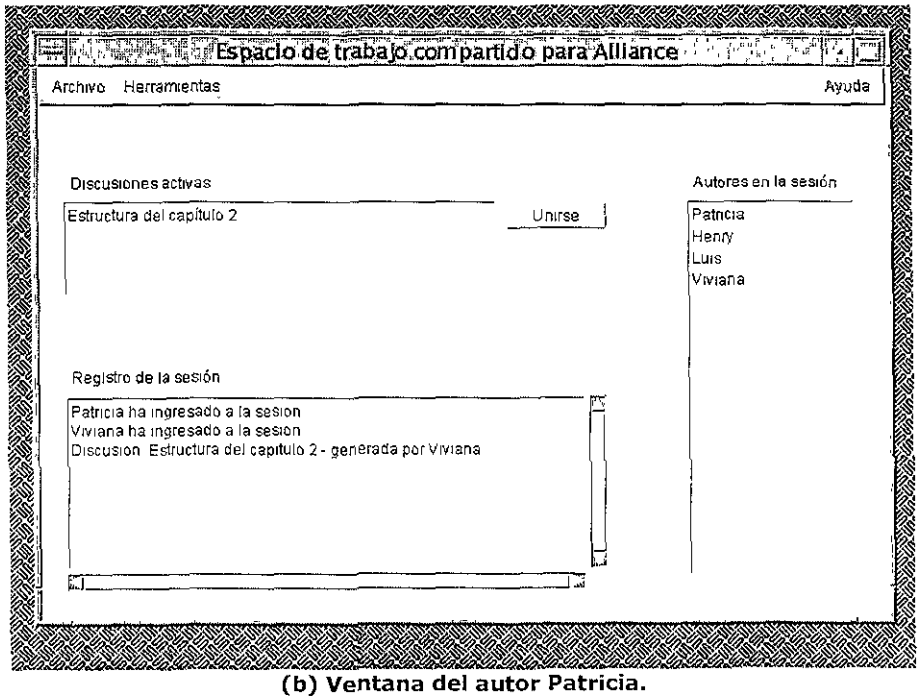
Para determinar el desempeño de la arquitectura definida para el espacio de trabajo, se utilizaron cuatro computadoras conectadas a Internet, en las cuales se midió el tiempo en el envío y la recepción de mensajes entre los autores y sus aportes a las discusiones.

4.5.1. Características

La funcionalidad del espacio de trabajo desarrollado se presenta mediante un ejemplo de una sesión de grupo, la cual involucra cuatro autores que utilizan las herramientas de colaboración:



(a) Ventana del autor Henry.



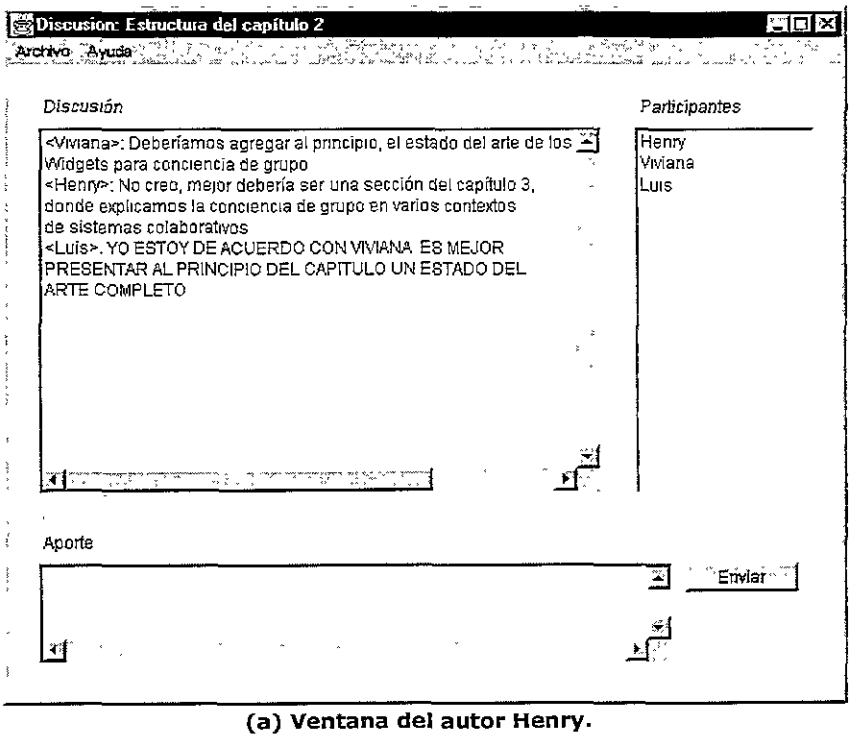
(b) Ventana del autor Patricia.

Figura 4.26. Ejemplo de dos ventanas principales del espacio de trabajo compartido.

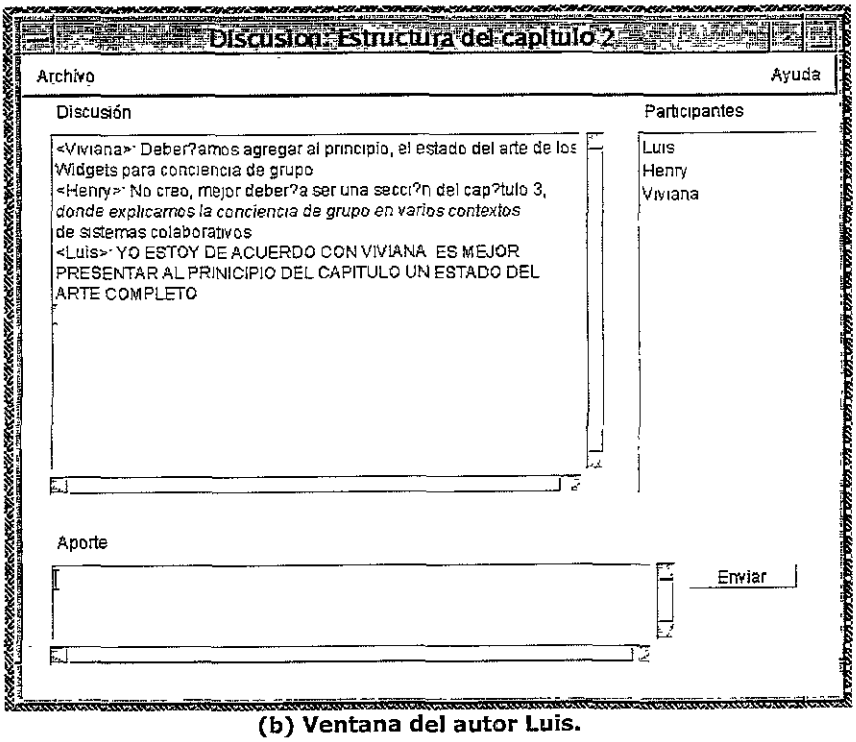
La figura 4.26 muestra dos ventanas principales del espacio de trabajo, en dos sistemas operativos diferentes, Windows en la parte (a) y Solaris en la parte (b). Nótese que ambas ventanas contienen la misma información sobre las discusiones en proceso.

Nótese que el contenido del registro de la sesión es diferente para cada autor, debido a que dicho registro se inicia en el momento en que se ingresa a la sesión. En la figura, el autor Henry ingresó al espacio de trabajo antes que el autor Patricia.

Por otra parte, en la figura 4.26 se muestra que las listas de los autores presentan la misma información pero en diferente orden. Esto sucede porque la lista muestra en primer lugar al autor al que corresponde cada ventana principal del espacio de trabajo.



(a) Ventana del autor Henry.



(b) Ventana del autor Luis.

Figura 4.27. Ejemplo de dos ventanas de discusión en la que participan tres de los cuatro autores en el espacio de trabajo compartido.

La figura 4.27 muestra dos ventanas de discusión sobre el tema **Estructura del capítulo 2**, la cual se lleva a cabo entre los autores Henry, Viviana y Luis. Nótese que las dos ventanas presentan el mismo contenido de la discusión. La lista de los participantes se presenta con diferente orden, ya que aparece en primer lugar el autor al que corresponde cada ventana.

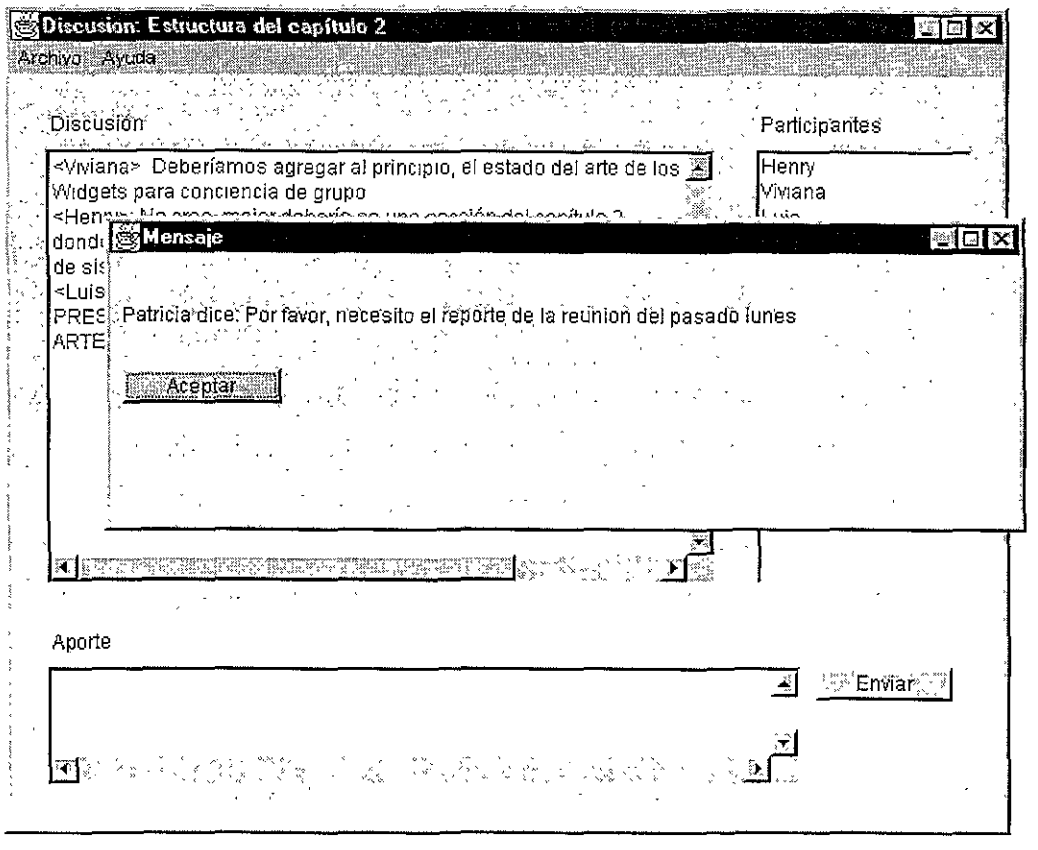
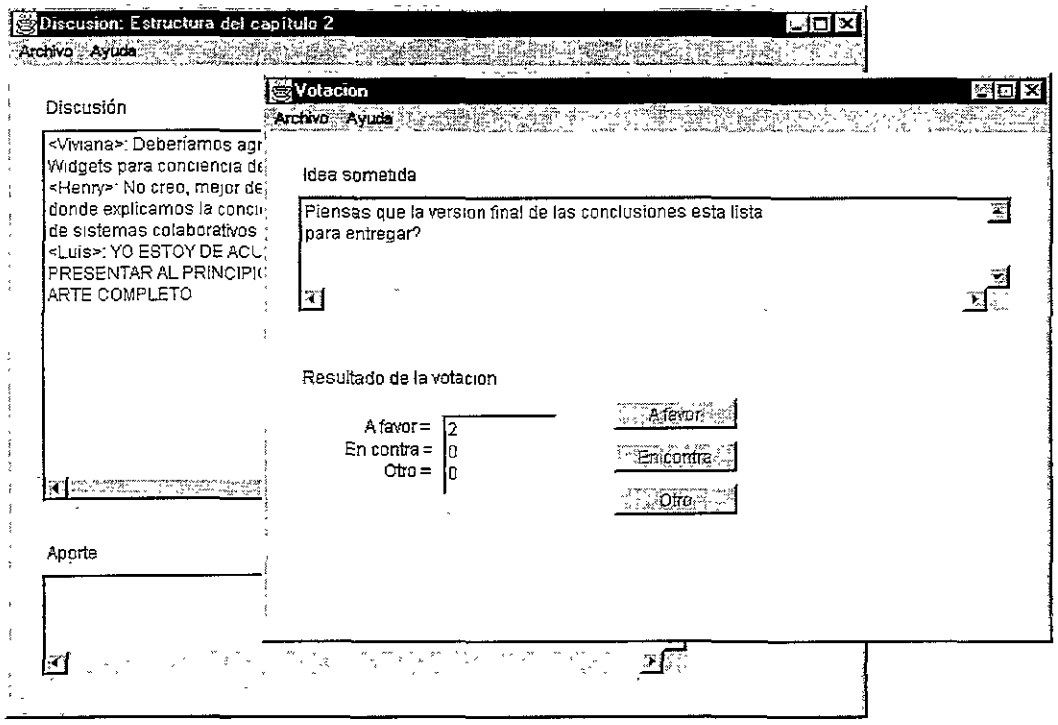


Figura 4.28. Ejemplo de un mensaje enviado por un autor específico.

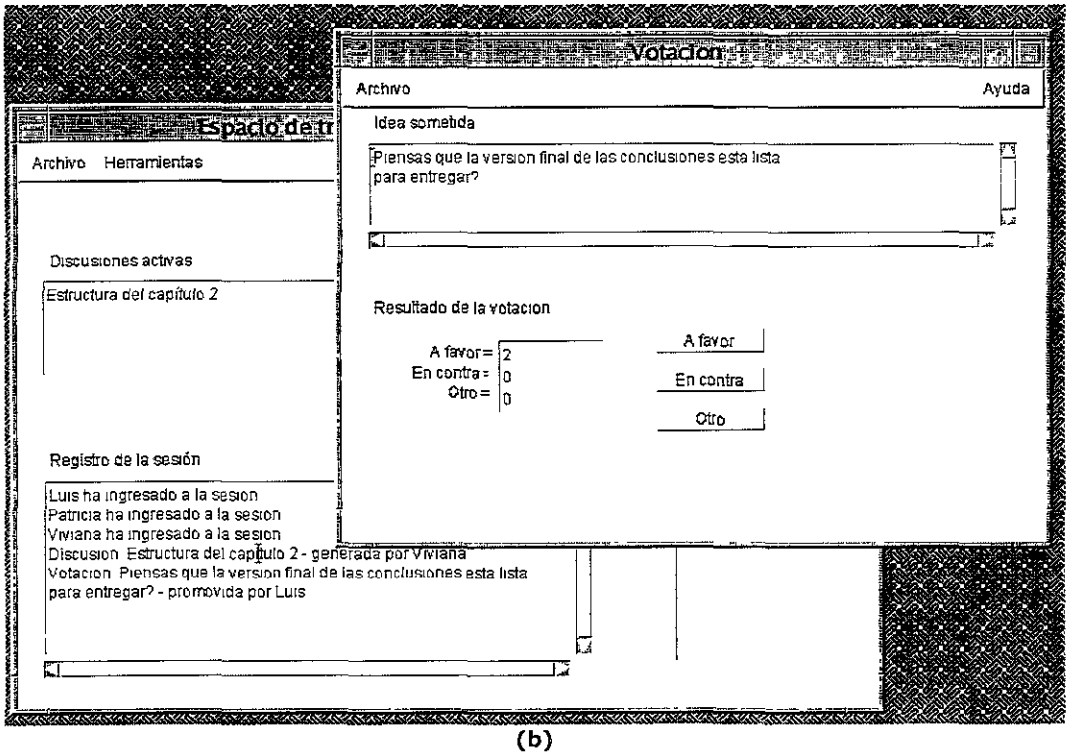
En la figura 4.28 se presenta un mensaje enviado por un autor específico. Nótese que la ventana de diálogo del mensaje, se despliega sobre las demás ventanas que se encuentran activas para los autores que reciben el mensaje. Esto ocurre debido a que el envío de mensajes promueve la interacción síncrona fuertemente acoplada.

Para finalizar el ejemplo de una sesión de grupo, en la figura 4.29 se presentan dos ventanas de una votación promovida sobre una idea en particular. Al igual que para el caso de los mensajes recibidos, estas ventanas se despliegan sobre las demás ventanas que se encuentren activas.

A diferencia de las ventanas de la discusión (figura 4.27), en las ventanas de la votación se muestra exactamente el mismo contenido para cada autor. Adicionalmente, el total de los votos para cada opción es percibido en forma síncrona y es actualizado en cada ventana sin que el autor tenga que intervenir en ello.



(a)



(b)

Figura 4.29. Ejemplo de dos ventanas de una votación promovida.

La figura 4.29 muestra que en la ventana de la parte (a) las opciones de votación no se encuentran habilitadas, porque el correspondiente autor ya emitió su voto. Por el contrario, en la ventana de la parte (b) se muestra que el autor aún no ha emitido su voto.

En el ejemplo presentado, se destaca que el espacio de trabajo facilita la interacción síncrona de dos maneras: débilmente acopada en el envío de eventos, fuertemente acoplada en la recepción de eventos.

Este ejemplo también permite identificar los elementos de conciencia de grupo (ver tabla 2.2) tenidos en cuenta para el diseño del espacio de trabajo: identidad (listas de usuario), acciones (registro de actividades) y esfera de influencia (a nivel de las herramientas de colaboración).

4.5.2. Desempeño de la arquitectura

Para medir el desempeño de la arquitectura definida para el espacio de trabajo, se utilizó un esquema de comunicación entre cuatro computadoras conectadas a Internet a través de medios diferentes. La información de estas computadoras es como sigue:

- Computadora Pentium II, sistema operativo Windows 98, conectada a Internet mediante red de área local a 10 Mbps. Dirección IP: 129.88.38.201. Ubicación: Grenoble, Francia.
- Computadora Pentium II, sistema operativo Windows 95, conectada a Internet mediante módem a 64 Kbps. Dirección IP: 200.65.68.165. Ubicación: domicilio particular, Ciudad de México.
- Computadora Pentium III, sistema operativo Linux RedHat 6.3, conectada a Internet mediante red de área local a 10 Mbps. Dirección IP: 192.168.1.2. Ubicación: Departamento de Ciencias de la Computación, IIMAS, UNAM.
- Estación de trabajo Sun Ultra 60, sistema operativo Solaris 7, conectada a Internet mediante red de área local a 10 Mbps. Dirección IP: 132.248.53.249. Ubicación: Edificio 5, Instituto de Ingeniería, UNAM.

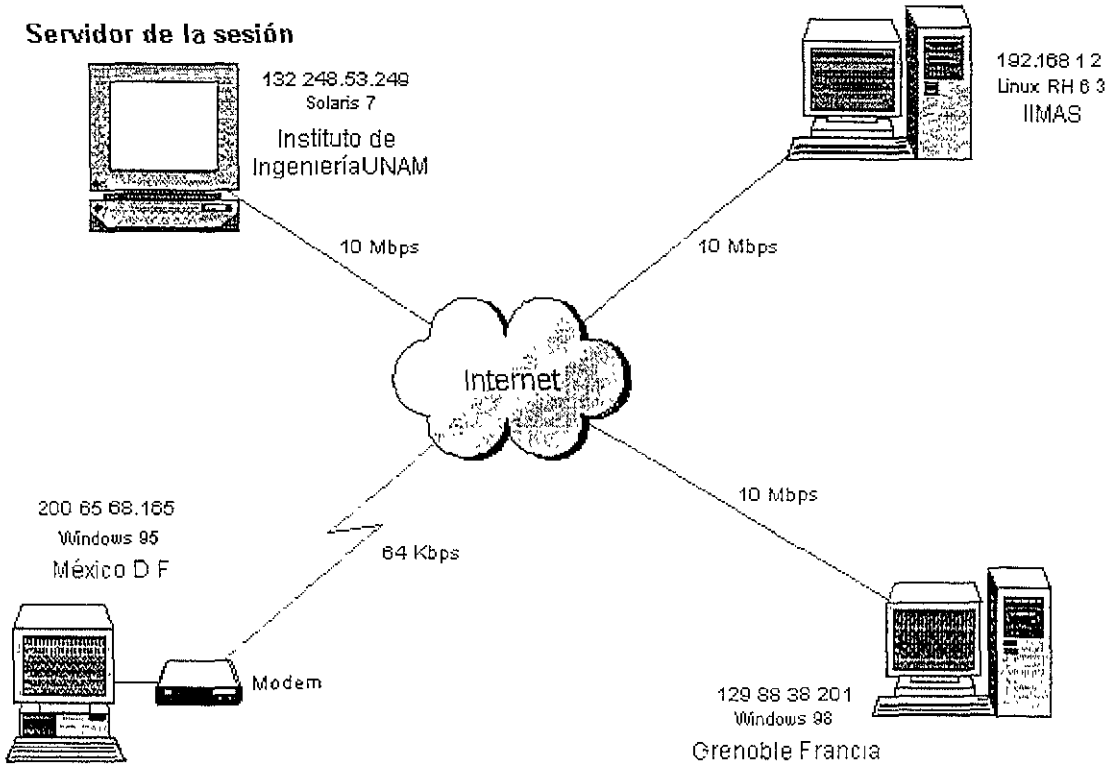


Figura 4.30. Esquema de conexión para la evaluación del desempeño de la arquitectura del espacio de trabajo compartido.

La figura 4.30 presenta el esquema de conexión utilizado para medir el desempeño de la arquitectura híbrida del espacio de trabajo. Aunque en este esquema de conexión se hicieron pruebas del funcionamiento de las tres herramientas de colaboración, para las medidas de tiempo no se utilizó la herramienta de votación, ya que esta herramienta no facilita una situación práctica para medir el tiempo de envío y recepción de un mensaje.

Desde cada computadora se realizó la conexión al servidor mediante una sesión configurada con sockets de TCP/IP, como protocolo de comunicación y con la computadora 132.248.53.249 designada como sitio servidor.

Las medidas de tiempo se tomaron considerando lo siguiente:

- En Java, cada caracter alfanumérico y especial ocupa 2 bytes. Con ello, se utilizó un tamaño promedio de 100 bytes para los aportes transmitidos en las discusiones y para los mensajes enviados. Estos tamaños no incluyen los encabezados que agrega el objeto Java Data (tamaño del mensaje, canal, emisor, etc.).

- El tiempo obtenido en cada computadora se midió a partir del momento en que cada autor envía un aporte o un mensaje, hasta que dicho autor recibe un aporte o mensaje de respuesta, de alguno de sus colegas. Este tiempo obtenido incluye el tiempo en que el autor destino recibe el mensaje o el aporte, lo lee, escribe su aporte o mensaje de respuesta y lo envía. Para simplificar los resultados, se tuvieron en cuenta los tiempos máximos obtenidos en el envío y recepción de aportes y mensajes, en cada computadora utilizada.

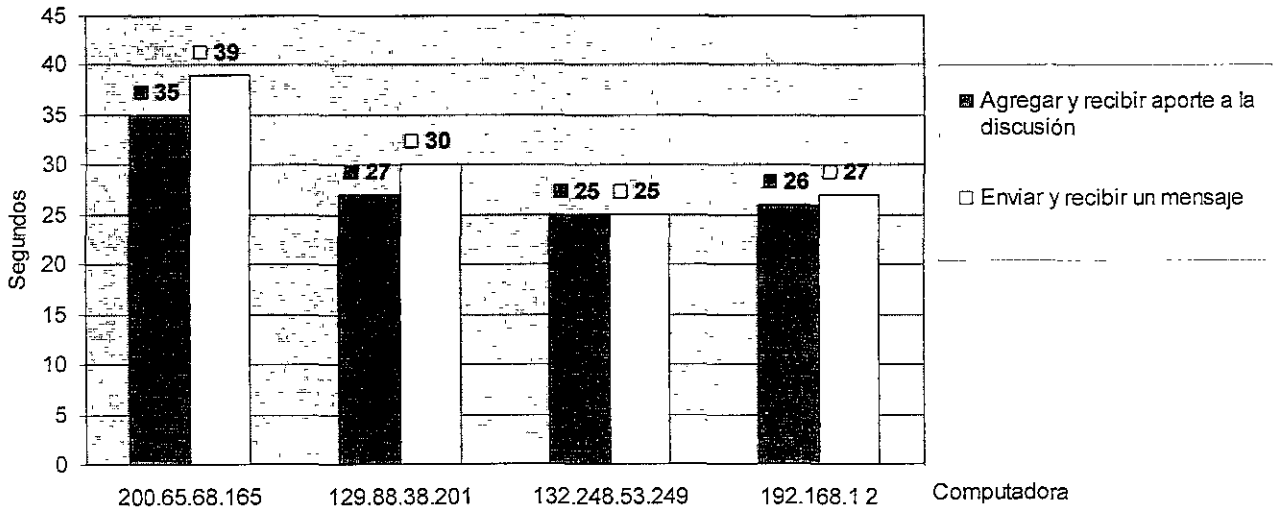


Figura 4.31. Comparación de tiempos en las cuatro computadoras utilizadas.

En la figura 4.31 se observa que el mayor tiempo obtenido corresponde a la computadora con la menor velocidad de conexión a Internet (64 Kbps). Este es un resultado lógico, porque el tiempo de envío y recepción de los mensajes, depende de la velocidad de la conexión a la red de comunicación. Sin embargo, la diferencia entre los tiempos obtenidos en esta computadora respecto a las demás, no excede de los 15 segundos, lo cual indica que todos los participantes interactúan en forma síncrona, con un promedio de tiempo de respuesta aceptable.

4.6. Discusión

En el presente capítulo se trató el desarrollo del espacio de trabajo compartido para el editor Alliance. Este espacio de trabajo proporciona las herramientas colaborativas síncronas necesarias para generar las discusiones, promover la votación y realizar el intercambio de mensajes durante una sesión de trabajo. Con ello, se apoya la fase de negociación que involucra el proceso de edición colaborativa de documentos en Alliance.

El espacio de trabajo utiliza el concepto de sesión de grupo para proporcionar una interface gráfica para los autores. Este concepto es suficiente para promover la conciencia de grupo síncrona en los elementos de identidad y de actividad, mediante la lista de usuarios en la sesión (en el espacio), las discusiones que se encuentran en proceso y un registro de las principales actividades (generar discusiones y promover votaciones) que cada autor realiza en la sesión.

Para que las herramientas del espacio de trabajo puedan ser usadas en forma colaborativa por los autores, se utiliza la difusión síncrona de eventos sobre canales confiables, mediante el método **sendToAll()** que provee el API JSDT. Este método utiliza un mecanismo asíncrono para la transmisión y entrega de los mensajes, el cual consiste en que cada cliente emisor verifica la recepción del mensaje con cada consumidor activo, después de que el mensaje se ha escrito en el canal. En ningún caso se presenta una negociación previa entre el emisor y los consumidores, para enviar y recibir en forma síncrona cada byte que compone el mensaje.

A pesar de utilizar un mecanismo asíncrono para la transmisión de mensajes, la difusión distribuida de eventos sobre canales independientes permite que en el espacio de trabajo se facilite la comunicación y la interacción en forma síncrona, fuerte y débilmente acoplada. Con ello, se apoya también la conciencia de grupo y se provee a los autores reunidos en la sesión, la sensación de estar trabajando en el mismo espacio al mismo tiempo.

La arquitectura híbrida definida a partir del API JSDT, se destaca principalmente por el manejo centralizado de la sesión de grupo y la difusión distribuida de mensajes y eventos, por parte de cada cliente. Además, esta arquitectura presenta las siguientes ventajas:

- Facilidad de construcción, ya que se usan librerías de clases de Java y del JSDT, que reducen considerablemente la programación de los sitios clientes y el servidor.
- La simplificación en las operaciones del servidor de la sesión, ya que éste actúa como un proceso intermediario en el envío de mensajes y mantiene la ubicación de los objetos de la sesión (clientes, canales, consumidores, etc.), para que cada cliente obtenga la referencia que requiere de ellos.
- La portabilidad del espacio de trabajo compartido, sobre diversas plataformas (Intel y RISC) y sistemas operativos (Windows, Solaris y Linux).
- La facilidad de agregar nuevos mecanismos de colaboración y nuevas herramientas, al espacio de trabajo, ya que las interfaces gráficas de usuario se encuentran completamente separadas del soporte para la comunicación. Con ello, el analista decide como utilizar dicho soporte, definiendo nuevos canales y consumidores, y como desplegar la información que se comparte a través de ellos.

Sin embargo, el control centralizado de la sesión y la dependencia de las referencias de los objetos creados en el servidor, ocasionan las siguientes desventajas de esta arquitectura:

- No se contempla la tolerancia a fallas, puesto que la falla o la desconexión del servidor, ocasiona la desconexión de los autores en el espacio de trabajo. A pesar de que cada cliente puede ejecutar los procesos del servidor, sólo uno de ellos puede hacerlo a la vez y

no existe un mecanismo de negociación, a nivel del control de la sesión, para que los clientes decidan quien asume el papel de servidor y como dirigir hacia éste, los objetos que se manejan como referencias.
- Aún cuando cada cliente puede crear los canales para las discusiones generadas y las votaciones promovidas, siempre se depende del sitio servidor para poder realizar esta operación. Esto se debe a que el servidor maneja en forma centralizada la información de los objetos de la sesión, por lo que al crear un nuevo canal, éste debe agregarse como un objeto de la sesión para que se le pueda hacer referencia desde lo demás clientes.

El procedimiento utilizado para medir el tiempo de respuesta del espacio de trabajo, es un procedimiento simple que no tiene en cuenta aspectos como la sincronización de las computadoras utilizadas o el análisis del tamaño de cada paquete TCP/IP transmitido utilizando sockets.

A pesar de ello, estas medidas arrojaron resultados satisfactorios, considerando que el tiempo obtenido incluye el tiempo en que cada autor realiza una acción, y que las computadoras utilizadas se encuentran ubicadas a grandes distancias. Con ello se concluye que la arquitectura híbrida definida para el espacio de trabajo utiliza en forma eficiente los sockets de TCP/IP sobre conexiones estables a Internet.

Finalmente, el presente capítulo ha mostrado una forma de utilizar parte de la metodología orientada a objetos unificada y el lenguaje UML, para el desarrollo de un sistema colaborativo en Internet. Con esta metodología se ha facilitado la identificación de las clases del dominio del problema (los mecanismos de colaboración en el espacio de trabajo) y las relaciones con las clases que se adoptan del lenguaje Java y del JSDT, para facilitar la implementación de su comportamiento.

Capítulo 5

Conclusiones y perspectivas

La presente investigación ha cumplido el objetivo de desarrollar un espacio de trabajo compartido, representado por medio de computadoras, para el editor Alliance. Este desarrollo enfatiza el apoyo a las actividades de la fase de negociación entre los autores, mediante herramientas síncronas que mejoran el proceso de edición de documentos en forma colaborativa.

Con el desarrollo obtenido en esta investigación se concluye que para apoyar dicha fase de negociación se requieren tres herramientas colaborativas claves: las discusiones simultaneas, la votación y el intercambio de mensajes. Estas herramientas no sólo facilitan la comunicación espontánea entre los autores, como sucede en el mundo real, sino que también promueven la conciencia de grupo en forma síncrona, beneficiando de manera directa el trabajo en conjunto.

5.1. Decisiones tomadas para alcanzar el objetivo propuesto

La edición de documentos en forma colaborativa involucra un alto porcentaje de actividades individuales, que los autores realizan de manera aislada, como por ejemplo la revisión o la escritura del documento. Sin embargo, para que este proceso de edición sea realmente eficiente, es fundamental que los autores puedan establecer comunicación directa entre ellos, con el fin de llevar a cabo las actividades que se derivan de esta comunicación, como lo son la negociación, la consolidación o la discusión.

Por ello, un editor colaborativo completo no sólo debe proporcionar herramientas que faciliten la interacción en forma asíncrona entre los autores, sino también en forma síncrona, para que puedan enterarse de las actividades y los eventos que generan sus colegas, en diferentes instantes de tiempo o en el instante en que suceden.

Los espacios de trabajo compartidos constituyen un medio apropiado para el uso de las herramientas síncronas y buscan establecer mecanismos que faciliten tanto la interacción humano-computadora, como la interacción humano-humano (frente a frente), asistida por computadora. Esto se sustenta con el hecho de que las representaciones de dichos espacios virtuales, adoptan completamente las características de los espacios de trabajo reales y resuelven la necesidad de que los participantes del trabajo en grupo perciban la sensación de estar trabajando e intercambiando su información, en un mismo espacio al mismo tiempo, en forma transparente a su ubicación física.

Las arquitecturas y herramientas actuales para el desarrollo de sistemas colaborativos en Internet ofrecen diversas soluciones para los problemas relacionados con la comunicación de grupos, la difusión y coordinación de eventos, la manipulación de objetos compartidos y, en un menor grado, el entendimientos de las actividades individuales y de grupo. En esta investigación se ha preferido utilizar un modelo híbrido para definir la arquitectura del espacio de trabajo para Alliance, porque dicho modelo facilita la combinación de los componentes centralizados y distribuidos según se requiera. Asimismo, el uso del lenguaje de programación

Java y el API Java Shared Data Toolkit no sólo simplifica el manejo de las sesiones de grupo, sino que agrega la portabilidad que hoy día es necesaria en cualquier tipo de sistema.

5.2. Desarrollo e implementación

El espacio de trabajo compartido desarrollado en esta investigación se sintetiza con las siguientes características:

- **Concepto utilizado**: sesión de grupo.
- **Formas de interacción**: síncrona, débilmente acoplada en el envío de eventos, fuertemente acoplada en la recepción de eventos.
- **Coordinación de eventos**: difusión de mensajes unicast y multicast. El control de la concurrencia de los eventos se apoya en el uso de múltiples canales. El ordenamiento de los mensajes se establece en forma centralizada, a través del orden en que se escriben en los canales de comunicación.
- **Interface gráfica**: vistas WYSIWIS de tamaño fijo.
- **Elementos de conciencia de grupo** : identidad, acciones (a nivel de las herramientas de discusión y votación) y esfera de influencia (a nivel de las herramientas de discusión y votación).
- **Mecanismos de conciencia de grupo**: comunicación directa mediante mensajes de texto.
- **Herramientas de conciencia de grupo**: listas de usuarios y registros de actividades en la sesión.

Durante la evaluación de este desarrollo se recopilaron las opiniones que los usuarios manifestaron sobre el funcionamiento general del espacio de trabajo. Estas opiniones se resumen como sigue:

- La facilidad de mantener varios espacios de comunicación síncrona, mediante las discusiones simultáneas, resulta ser una característica bastante útil, ya que se pueden tratar diversas ideas al mismo tiempo y en forma independiente. Asimismo, el envío de mensajes a un usuario en particular, resulta muy apropiado para establecer una forma de comunicación privada, especialmente cuando se participa en una discusión de grupo.

- La principal desventaja encontrada por los usuarios se relaciona con la característica de percepción de los eventos de manera síncrona fuertemente acoplada. En los procesos formales como el de la edición colaborativa, es necesario evitar que los autores sean interrumpidos en sus actividades. Con ello, se sugiere que los mensajes no se desplieguen como una ventana de diálogo, sino que se almacenen en un espacio predefinido, desde el cual se notifique al autor la ocurrencia del evento, por medio de un sonido, un cambio de color o un cambio del estado de dicho espacio.

Asimismo, las medidas de tiempo obtenidas en la evaluación de las herramientas discusión y envío de mensajes (ver figura 4.31), permiten concluir que el espacio de trabajo mantiene un tiempo de respuesta aceptable, aún en conexiones a grandes distancias, lo cual estimula la participación de los usuarios en las actividades de negociación, requeridas no sólo en la edición colaborativa sino en diferentes contextos de trabajo en grupo.

Con el desempeño y el tiempo de respuesta obtenido en el espacio de trabajo compartido para Alliance, se concluye que la arquitectura híbrida definida utiliza en forma eficiente las capacidades del JSDT, para el intercambio de la información sobre canales de comunicación confiables y los sockets de TCP/IP sobre conexiones estables a Internet.

5.3. Contribución de la tesis

En el presente trabajo de investigación se logró un desarrollo teórico y práctico que contribuye a la infraestructura del ambiente multipropósito para la edición colaborativa en Internet. Este desarrollo aporta elementos que pueden tomarse como base para nuevos desarrollos de mecanismos y herramientas, que exploren nuevas alternativas para facilitar el trabajo colaborativo y su aplicación.

El desarrollo teórico comprende la recopilación de un estado del arte de la problemática actual de los espacios de trabajo compartidos y las tendencias que existen en los principales desarrollos dentro de CSCW y HCI. Con este estado del arte se proporcionan descripciones y referencias importantes de trabajos que pueden orientar nuevos proyectos de investigación.

El desarrollo práctico comprende las experiencias obtenidas en el análisis, el diseño y la construcción de un sistema colaborativo síncrono, utilizando la metodología orientada a objetos unificada y el lenguaje UML, las cuales pueden servir como guía para modelar otro tipo de problemas comunes en los sistemas distribuidos, como lo son entre otros, el control de la concurrencia, la consistencia de la información y la distribución del procesamiento.

5.4. Limitaciones

A pesar de que el concepto de sesión, en el cual se basa el espacio de trabajo compartido, es suficiente para proveer las herramientas colaborativas desarrolladas, dicho concepto limita la representación de los siguientes elementos de conciencia de grupo: ubicación, nivel de actividad, acciones que se llevan a cabo fuera de las herramientas colaborativas, intenciones, extensión de la información que se puede percibir en el espacio de trabajo y las expectativas para facilitar la anticipación de las actividades en grupo.

La interacción dentro del espacio de trabajo depende del uso de las herramientas de colaboración, es decir, los autores sólo colaborar entre sí, cuando realizan discusiones, votaciones o intercambian mensajes. El espacio de trabajo carece de otros elementos como un depósito común de objetos compartidos o un pizarrón colaborativo, que incrementen las posibilidades de interacción entre los autores.

La arquitectura híbrida definida presenta un desempeño aceptable, aún en situaciones donde los autores se encuentran ubicados a gran distancia. Sin embargo, la dependencia total de los objetos del servidor y el manejo centralizado de la sesión, implican que no se soporte la tolerancia a fallas o desconexiones del servidor. Además, para el funcionamiento adecuado del espacio de trabajo, se requieren conexiones estables entre los clientes y el servidor, y no se contemplan mecanismos de recuperación en caso de que falle alguno de ellos.

Dado que las discusiones y las votaciones constituyen los principales objetos de colaboración dentro del espacio, la persistencia de dichos objetos se reduce al almacenamiento en forma local de su contenido. Sin embargo, la persistencia del espacio mismo no está contemplada, por ejemplo en situaciones donde los autores requieran retomar una discusión que se llevó a cabo en sesiones de grupo anteriores.

5.5. Perspectivas

Como ya se ha mencionado, el espacio de trabajo compartido para Alliance constituye un desarrollo enmarcado dentro del proyecto **Ambiente Multipropósito para la Edición Colaborativa en Intranet e Internet**, del Departamento de Ciencias de la Computación del IIMAS. Las características de este espacio de trabajo están siendo evaluadas actualmente en

el proyecto **Edición coLaborativa de documentos XML en Internet (ELXI)**, con el fin de adoptar las herramientas síncronas de colaboración que apoyan la negociación durante la edición de documentos.

Es por ello que para el proyecto ELXI se establecen las siguientes recomendaciones y sugerencias:

- Extender el concepto de sesión a otros conceptos como el de salones y escritorios, que incrementen las capacidades de interacción entre los autores, la diversidad de los objetos de colaboración y se ajusten a las tendencias de los espacios de trabajo compartidos en Internet.
- Ampliar las capacidades del espacio de trabajo con una herramienta para el depósito de objetos compartidos (documentos, imágenes, archivos de audio y video, URLs, etc.), que se agregue como una nueva interface gráfica de ventana. Esta herramienta, debe también extender la arquitectura híbrida definida para el almacenamiento distribuido de los objetos compartidos. De acuerdo con las características del espacio de trabajo y del API JSDT, dicha herramienta debe establecer sus propios canales de comunicación y utilizar los métodos de la clase Channel para el intercambio de los objetos.
- En relación con el manejo centralizado de la sesión de grupo, se recomienda que se defina un servicio **Registry** distribuido, que proporcione las referencias a los objetos de la sesión (canales, clientes, consumidores, etc.). Este servicio debe ser independiente de la ubicación, es decir, las referencias que se crean en los clientes, para los objetos de la sesión, no deben depender del lugar donde dichos objetos fueron creados. Cabe mencionar que esta propuesta también implica una modificación de la arquitectura híbrida definida, para que durante la creación y la conexión a la sesión, no se utilice un objeto sesión creado en un servidor, sino un objeto sesión replicado en diversos sitios servidores de sesión.
- A pesar de que se obtuvieron resultados satisfactorios con el uso de sockets de TCP/IP, se necesita experimentar con el desarrollo de herramientas que utilicen sesiones HTTP o LRMP. Con ello, se puede extender la arquitectura definida para que se integre a la Web, proporcionando las ventajas de la comunicación y la interacción síncrona, hacia nuevos requerimientos como los que propone el proyecto AllianceWeb.
- Realizar una evaluación completa del uso del espacio de trabajo compartido, en un ambiente real de producción de documentos en Alliance. Con ello, se deben establecer nuevos requerimientos que sustenten el desarrollo de nuevos mecanismos de colaboración y mejoren los desarrollados en esta investigación.

Los espacios de trabajo representados por computadoras constituyen un medio apto para abolir la distancia entre los participantes del trabajo colaborativo, independientemente del dominio de la tarea que se realiza en conjunto. Sin embargo, el futuro de estas representaciones depende principalmente de los avances tecnológicos de las infraestructuras de comunicación y de la definición y la aplicación de mejores mecanismos que se ajusten a la forma como las personas interactúan en el mundo real.

En CSCW existe la tendencia de que los espacios de trabajo compartidos evolucionen hacia verdaderos ambientes virtuales colaborativos y las representaciones del espacio, las personas y los objetos, se apoyen en la realidad virtual para simular con mayor precisión los espacios de trabajo reales. Sin embargo, el principal problema que persiste en estas tendencias, consiste en que a medida que aumenta la capacidad de la representación del ambiente virtual, aumentan los requerimientos de procesamiento y la cantidad de información que debe intercambiarse para facilitar la interacción síncrona sobre un medio como Internet.

Bibliografía

[App87] Apple Computer Inc. **Human Interface Guidelines: The Apple Desktop Interface**. Addison Wesley, 1987. pp. 144.

[Bae93] Ronald Baecker, Dimitrios Nastos, Ilona Posner y Kelly Mawby, **The User-centred Iterative Design Of Collaborative Writing Software**. *Proceedings on Human Factors in Computing Systems,* Amsterdam Holanda, abril de 1993. pp. 399-405.

[Bae94] Ronald Baecker, Geof Glass, Alex Mitchell e Ilona Posner, **SASSE: the Collaborative Editor**. *Video presentado en ACM Conference on Human Factors in Computing Systems*, 1994, transcripción disponible en **http://www.dgp.toronto.edu/CMRG/Projects/CWPublications/video.html**.

[Beg97] James Begole, Craig Struble y Clifford Shaffer, **Transparent Sharing of Java Applets: A Replicated Approach**. *Proceedings of the 1997 ACM Symposium on User Interface Software and Technology (UIST'97),* N.Y. USA. pp. 55-64.

[Ben95] Richard Bentley, Thilo Horstmann, Klaas Sikkel y Jonathan Trevor, **Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System**. *The World Wide Web Journal: Proceedings of the 4th International WWW Conference*, Boston USA, diciembre de 1995. pp. 63-74.

[Ber94] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen y Arthur Secret, **The World-Wide Web**. *Communications of the ACM*, volumen 37(8), agosto de 1994. pp. 76-82.

[Ber96] Philip Bernstein, **Middleware: a Model for Distributed System Services**. *Communications of the ACM*, volumen 39(2), febrero de 1996. pp. 86-98.

[Berl96] Dan Berlin et al., **CGI Programming**. Primera Edición, Sams.net Publishing, 1996. pp. 620.

[Bla91] Uyless Black, **OSI a Model for Computer Communications Standards**. Prentice Hall, 1991. pp. 528.

[Boo99] Grady Booch, James Rumbaugh e Ivar Jacobson, **The Unified Modeling Language User Guide**. Addison-Wesley, 1999. pp. 482.

[Bur99] Rich Burridge, **Java Shared Data Toolkit User Guide**. Sun Microsystems, octubre de 1999. Disponible en: **http://java.sun.com/products/java-media/jsdt/index.html**.

[Cou95] George Coulouris, Jean Dollimore y Tim Kindberg, **Distributed Systems, Concepts and Design**. Segunda Edición, Addison Wesley, 1995. pp. 644.

[Dec99] Dominique Decouchant, Ana María Martínez y Esther Martínez, **AllianceWeb: Cooperative Authoring on the WWW**. *Proceedings of the CRIWG'99*, Cancún México. pp. 286-295.

[Dec01] Dominique Decouchant, Jesús Favela y Ana M. Martínez, **PIÑAS: A Middleware for Web Distributed Cooperative Authoring**. *Proceedings of SAINT'2001, The 2001 Symposium on Applications and the Internet*, San Diego CA USA, 8 al 12 de enero del 2001 (en prensa).

[DiBo99] Márcio Dias y Marcos Borges, **Development of Groupware Systems with the COPSE Infraestructure**. *Proceedings of the CRIWG'99*, Cancún México. pp. 278-285.

[DoBe92] Paul Dourish y Victoria Bellotti, **Awareness and Coordination in Shared Workspaces**. *Proceedings of the CSCW'92,* Toronto Canadá. pp. 107-114.

[DoBl93] Paul Dourish y Sara Bly, **Portholes: Supporting Awareness in a Distributed Work Group**. *Readings in Groupware and Computer-Supported Cooperative Work Assisting Human-Human Collaboration*, Ronald Baecker (editor), Morgan Kaufmann Publisher, 1993. pp. 809-815.

[ElGi89] Clarence Ellis y Simon Gibbs, **Concurrency Control in Groupware Systems**. *Proceedings of the ACM SIGMOD'89*, Portland OR USA. pp. 399-407.

[Ell91] Clarence Ellis, Simon Gibbs y Gail Rein, **Groupware, Some Issues and Experiences**. *Communications of the ACM*, volumen 34(1), enero de 1991. pp. 38-58.

[Fis88] Robert Fish, Robert Kraut, Mary Leland y Michael Cohen, **Quilt: A Collaborative Tool for Cooperative Writing**. *Conference ACM SIGOIS and IEEECS TC-OA on Office Information Systems*, Palo Alto CA USA, marzo de 1988. pp. 30-37.

[FrAv98] Emmanuel Frécon y Anneli Avatare, **Building Distributed Virtual Environments to Support Collaborative Work**. *ACM Symposium on Virtual Reality Software and Technology (VRST'98)*, Taipei Taiwan. pp. 105-114.

[Fre99] Eric Freeman, Susanne Hupfer y Ken Arnold, **JavaSpaces Principles, Patterns and Practice**. *The Jini Technology Series*. Primera edición, Addison-Wesley, 1999. pp. 344.

[GaHa97] Ulrich Gall y Franz J. Hauck, **Promondia: A Java-Based Framework for Real-time Group Communication in the Web**. *Sixth International WWW Conference*, California USA, abril de 1997.

[Gre00] Saul Greenberg, **Real Time Distributed Collaboration**. *Encyclopedia of Distributed Computing*, P. Dasgupta y J. E. Urban (editores), Kluwer Academic Publishers, 2000 (en prensa).

[GrMa94] Saul Greenberg y David Marwood, **Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface**. *Proceedings of the ACM CSCW'94*, Chapel Hill USA. pp. 207-217.

[GrRo99] Saul Greenberg y Mark Roseman, **Groupware Toolkits for Synchronous Work**. *Computer-Supported Cooperative Work*, M. Beaudouin-Lafon (editor), John Wiley & Sons, 1999. pp. 135-168.

[GuFu99] Luis Guerrero y David Fuller, **Design Patterns for Collaborative Systems.** *Proceedings of the CRIWG'99*, Cancún México. pp. 270-277.

[GuGr99] Carl Gutwin y Saul Greenberg, **A Framework of Awareness for Small Groups in Shared-Workspace Groupware.** *Technical Report 99-2*, Departamento de Ciencias de la Computación, Universidad de Saskatchewan, Canada, 1999. Disponible en **http://www.cs.usask.ca/faculty/gutwin/publications.html.**

[Gut96a] Carl Gutwin, Saul Greenberg y Mark Roseman, **Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation.** *Proceedings of the HCI'96*, Londres U.K. pp. 281-298.

[Gut96b] Carl Gutwin, Mark Roseman y Saul Greenberg, **A Usability Study of Awareness Widgets in a Shared Workspace Groupware System.** *Proceedings of ACM CSCW'96 Conference on Computer Supported Cooperative Work*, Boston USA. pp. 258-267.

[HaWi92] Jörg M. Haake y Brian Wilson, **Supporting Collaborative Writing of Hyperdocuments in SEPIA.** *Proceedings of the CSCW'92,* Toronto Canadá. pp. 138-146.

[HoBe97] Thilo Horstmann y Richard Bentley, **Distributed Authoring on the Web with the BSCW Shared Workspace System.** *ACM StandardView*, volumen 5(1), marzo de 1997. pp. 9-16.

[HoKi97] Markus Horstmann y Mary Kirtland, **DCOM Architecture.** *Reporte Técnico,* Instituto de Ciencias de la Computación Universidad Tecnológica de Warsaw, Polonia, julio de 1997. Disponible en **http://cyclop.ii.pw.edu.pl/sr/DCOMArchitecture/dcomarch.htm.**

[Ing95] David Ingham, Mark Little, Steve Caughey y Santosh Shrivastava, **W3Objects: Bringing Object-Oriented Technology to the Web.** *Proceedings of the Fourth International World Wide Web Conference*, Boston USA, diciembre de 1995.

[Ise98] Philip Isenhour, **Sieve: A Java-Based Framework for Collaborative Component Composition**. Tesis de Maestría en Ciencias de la Computación del Instituto Politécnico de Virginia, Blacksburg Viginia USA, febrero de 1998.

[Kin96] Tim Kindberg, **Mushroom: A Framework for Collaboration and Interaction across the Internet**. *ERCIM Workshop on CSCW and the Web*, Sankt Augustin Alemania, febrero de 1996.

[Kin97] Tim Kindberg, **An Open Architecture for Replicated Shared Objects**. *The PerDiS Workshop on Persistence and Distribution in Java*, Lisbon Portugal, octubre de 1997.

[SoCh94] Markus Sohlenkamp y Greg Chwelos, **Integrating Communication, Cooperation and Awareness: The DIVA Virtual Office Environment.** *Proceedings of the Conference CSCW'94*, Chapel Hill USA. pp. 331–343.

[Ste87] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning y D. Tatar, **WYSIWIS Revised: Early Experiences with Multiuser Interfaces**. *ACM Transactions on Information Systems*, volumen 5(2), 1987. pp. 147-167.

[TaKi92] Haruo Takemura y Fumio Kishino, **Cooperative Work Environment Using Virtual Workspace**. *Proceedings of the CSCW'92,* Toronto Canadá. pp. 226-232.

[Tre97] Jonathan Trevor, Thomas Koch y Gerd Woetzel, **MetaWeb: Bringing Synchronous Groupware to the World Wide Web**. *Proceedings of the European Conference on Computer Supported Cooperative Work (ECSCW'97)*, Lankaster U.K.

[Wal96] Michael Walther, **Supporting Development of Synchronous Collaboration Tools on the Web with GroCo**, *Proceedings of the ERCIM Workshop on CSCW and the Web*, Sankt Augustin Alemania, septiembre de 1996.

[Wan00] Nanbor Wang, Douglas Schmidt y Carlos O'Ryan, **An Overview of the CORBA Component Model**. *Component-Based Software Engineering: Putting the Pieces Together*, G. Heineman y W. Councill (editores), Addison-Wesley, 2000 (en prensa).

[Whi97] E. James Whitehead, **World Wide Web Distributed Authoring and Versioning (WebDAV): An Introduction**. *ACM StandardView*, volumen 5(1), marzo de 1997. pp. 3-8.

[Woo99] Alan Wood, **Coordination with Attributes**. *Coordination Languages and Models: Proceedings of the Third International Conference COORDINATION '99*, Amsterdam Holanda. pp. 21-36.

Apéndice 1. Especificaciones de implementación

Clase:	**WSCliente.java**
Descripción:	Implementa la interface Client del JSDT, definiendo la variable "nombre" que será asignada a cada cliente y el método getName para regresar dicha variable a sus instancias.
Paquetes: usados	com.sun.media.jsdt.*;
Atributos:	String nombre,

Métodos:	// Constructor
	public WSCliente(String nombre) {
	this.nombre = nombre;
	}
	
	// Método que autentica el objeto para que sea manejable. En el espacio de trabajo no es
	// necesario. En este caso regresa null.
	
	public Object authenticate(AuthenticationInfo info) {
	System.err.println("WSCliente: autenticación");
	return(null);
	}
	
	// Regresa el nombre del cliente
	public String getName() {
	return(nombre);
	}

Clase:	**WSServidor.java**
Descripción:	Crea la sesión "WSSesion" y los canales WSCanal, EVCanal, LUCanal, LDCanal. Verifica que el Registry está iniciado o de lo contrario lo inicia.
Paquetes : usados	com.sun.media.jsdt.*;
Atributos:	String nombreSesion = "WSSesion"; WSCliente cliente; Session WSSesion; URLString url; String tipoSesion="socket"; String nombrehost; int puertohost;

Métodos:	public static void main(String args[]) {
	url = URLString.createSessionURL(nombrehost, puertohost, tipoSesion, nombreSesion);
	
	// Está iniciado el Registry ? Si no, iniciarlo
	
	if (RegistryFactory.registryExists(tipoSesion) == false) {
	RegistryFactory.startRegistry(tipoSesion);
	}
	// Crea la sesión y los canales
	cliente = new WSCliente("Servidor");
	WSSesion = SessionFactory.createSession(cliente, url, false);
	WSSesion.createChannel(cliente, "WSCanal", true, true, false);
	WSSesion.createChannel(cliente, "LUCanal", true, true, false);
	WSSesion.createChannel(cliente, "LDCanal", true, true, false);
	WSSesion.createChannel(cliente, "EVCanal", true, true, false);
	}

Clase:	**WSConsumidor.java**
Descripción:	Implementa la interface ChannelConsumer para el registro de actividades en la sesión. Recibe los mensajes enviados por el canal WSCanal e implementa el método dataReceived (propio de la interface), el cual notifica al cliente cuando se han recibido los mensajes.
Paquetes: usados	com.sun.media.jsdt.*; java.awt.*;

Atributos:	// Nombre del consumidor del canal
	protected String nombre;
	// Ubicación para escribir los mensajes recibidos, relacionados con las actividades en la sesión
	TextArea areamensajes;

Métodos:	// Constructor
	public WSConsumidor(String nombre, TextArea areamsg) {
	this.nombre = nombre;
	this.areamensajes = areamsg;
	}
	
	// Implementación del método dataReceived. Es sincronizado para evitar que llamadas
	// subsecuentes del método, sobreescriban los datos de las llamadas previas.
	
	public synchronized void dataReceived(Data datos) {
	int posicion = 0;
	String datosenviados = datos.getDataAsString();
	
	// agrega el mensaje al final del área de mensajes
	posicion = areamensajes.getText().length();
	areamensajes.insert(datosenviados, posicion);
	}

Clase:	**EVConsumidor.java**
Descripción:	Implementa la interface ChannelConsumer para obtener los eventos difundidos en la sesión, los cuales comprenden votación promovida, invitación a discusión y mensajes entre los autores. Recibe los mensajes enviados por el canal EVCanal e implementa el método dataReceived().
Paquetes y clases usadas:	com.sun.media.jsdt.*; java.awt.*; java.lang.String;

Atributos:	// Nombre del consumidor del canal
	protected String nombre;
	// Ubicación para escribir los mensajes recibidos, relacionados con los eventos en la sesión
	TextArea idea;
	String msg;
	// tipo de mensaje recibido
	int tipo;

Métodos:	// Constructor
	public EVConsumidor {}
	
	// dataReceived.
	public synchronized void dataReceived(Data datos) {
	
	// verifica el tipo de mensaje recibido, para ello valida el primer caracter del mensaje. El resto
	// del mensaje contiene la idea sometida o el tema de la discusión o el mensaje entre autores.
	// Si el primer caracter del mensaje es
	// 0, crea y despliega la interface Votación, con la idea sometida
	// 1, despliega la ventana de diálogo (EVDialogo) con la invitación a la discusión y el tema
	// 2 o 3, despliega la ventana de diálogo (EVDialogo) con el mensaje recibido y su emisor.
	}

Clase:	**LDConsumidor.java**
Descripción.	Implementa la interface ChannelConsumer para obtener la lista de discusiones en proceso. Recibe los mensaje enviados por el canal LDCanal e implementa el método dataReceived(), en el cual se verifica si se agrega o se retira la discusión de la lista.
Paquetes: usados	com.sun.media.jsdt.*; java.awt.*;

Atributos:	// Nombre del consumidor del canal
	protected String nombre;
	// Ubicación para escribir los mensajes recibidos, relacionados con las discusiones en proceso
	List discusiones;

Métodos:	// Constructor
	public LDConsumidor(String nombre, List discs) {
	this.nombre=nombre;
	this.discusiones=discs;
	}
	
	// dataReceived.
	public synchronized void dataReceived(Data datos) {
	
	// valida el primer caracter del mensaje. El resto del mensaje es el tema de la discusión.
	// Si el primer caracter del mensaje es
	// 0, agrega el tema a la lista discusiones
	// 1, retira el tema de la lista discusiones
	}

Clase:	**LUConsumidor.java**
Descripción:	Implementa la interface ChannelConsumer para obtener la lista de autores en la sesión. Recibe los mensajes enviados por el canal LUCanal e implementa el método dataReceived(), en el cual se verifica si se agrega o se retira al autor de la lista.
Paquetes: usados	com.sun.media.jsdt.*; java.awt.*;

Atributos:	// Nombre del consumidor del canal
	protected String nombre;
	// Ubicación para escribir los mensajes recibidos, relacionados con los autores en la sesión
	List autores;

Métodos.	// Constructor
	public LUConsumidor(String nombre, List auts) {
	this.nombre=nombre;
	this.autores=auts;
	}
	
	// dataReceived.
	public synchronized void dataReceived(Data datos) {
	
	// valida el primer caracter del mensaje. El resto del mensaje es el nombre del autor.
	// Si el primer caracter del mensaje es
	// 0, agrega el autor la lista autores
	// 1, retira el autor de la lista autores
	}

Clase:	**WSUsuario.java**
Descripción.	Realiza el proceso general de conexión a la sesión y a los canales WSCanal, LUCanal y LDCanal. Realiza el proceso de desconexión de la sesión y los canales
Paquetes: usados	com.sun.media.jsdt.*; java.awt.*;
Atributos:	String nombre, nombrehost, url; Session sesion, Channel canal, canal_lusuarios, canal_ldiscusiones; int puertohost, String tiposesion="socket"; TextArea logsesion, String nombresesion="WSSesion"; WSConsumidor wsConsumidor; LUConsumidor luConsumidor; LDConsumidor ldConsumidor; Data datos; WSCliente wsCliente; List listausuarios, discusiones,

Métodos:	// Conectar
	private void conectar() {
	url = URLString.createSessionURL(nombrehost, puertohost, tiposesion, nombresesion);
	cliente = new WSCliente(nombre);
	sesion = SessionFactory.createSession(cliente, url, true);
	canal = sesion.createChannel(cliente, "WSCanal", true, true, true);
	wsConsumidor = new WSConsumidor(cliente.getName(), logsesion);
	canal.addConsumer(cliente, wsConsumidor);
	canal_lusuarios = sesion.createChannel(cliente, "LUCanal", true, true, true);
	luConsumidor = new LUConsumidor(cliente.getName(), listausuarios);
	canal_lusuarios.addConsumer(cliente, luConsumidor);
	canal_ldiscusiones = sesion.createChannel(cliente, "LDCanal", true, true, true);
	ldConsumidor = new LDConsumidor(cliente getName(), discusiones);
	canal_ldiscusiones.addConsumer(cliente, ldConsumidor);
	// difundir el evento de autor conectado al canal_lusuarios y a canal
	datos = new Data(nombre);
	datos setPriority(Channel.HIGH_PRIORITY);
	canal_lusuarios.sendToAll(cliente, datos);
	datos = new Data(nombre+" ha ingresado a la sesión\n");
	datos.setPriority(Channel HIGH_PRIORITY);
	canal.sendToAll(cliente, datos);
	}
	// Desconectar
	private void desconectar {
	// retirar al cliente de los canales y de la sesión.
	}

Clase:	**Discusion.java**
Descripción:	Realiza el proceso general de la discusión y construye la respectiva ventana (Frame). Implementa las interfaces WindowListener y ActionListener para el control de los eventos que ocurren en la interface. Define el método para crear o conectarse al canal de la discusión y de la lista de los autores que participan en ella.
Paquetes: usados	com.sun.media.jsdt.*;java.awt.event.*; java.awt.*;
Atributos:	WSUsuario wsUsuario; DiscConsumidor discConsumidor; Data datos; Channel canal, canal_lusud; LUDConsumidor ludConsumidor; // variables de la interface gráfica (botones, menú, listas, área de mensajes, etc.)

Métodos	// Constructor
	Discusion(Frame ventana, String tema, String usuario, WSCliente cliente, Session sesion) {
	// crear el menú y la interface gráfica
	// agregar los objetos que realizan una acción utilizando "objeto.addActionListener"
	// llama el método conectar() (declarado más abajo)
	// llama el método listaautores() (declarado más abajo)
	}
	public void actionPerformed(ActionEvent event){
	// verificar el objeto que realiza la acción. Dependiendo del objeto:
	// llamar el método escribirmensaje(String mensaje) (declarado más abajo)
	}
	private void conectar(){
	// crea un nuevo canal para la discusión o se conecta si ya existe. El canal creado se almacena en el objeto canal
	// crea un nuevo canal para la lista de autores o se conecta si ya existe.
	// crea y agrega los objetos consumidores de las clases DiscConsumidor y LUDConsumidor
	}
	private void listaautores() {
	// crea y despliega la lista de autores en la discusión, consultando los consumidores del canal canal_lusud
	}
	private void escribirmensaje(String msg) {
	// Escribe el mensaje de aporte a la discusión en el canal canal
	}
	private void guardadiscusion() {
	// Escribe los aportes de la discusión en un archivo de texto
	}

Clase:	**DiscConsumidor.java**
Descripción:	Implementa la interface ChannelConsumer para obtener los aportes enviados a cada discusión. Recibe los mensajes enviados por el canal de la discusión e implementa el método dataReceived. Por cada canal que se crea para cada discusión, se crea un objeto de la clase DiscConsumidor.
Paquetes: usados	com.sun.media.jsdt.*; java.awt.*;

Atributos:	// Nombre del consumidor del canal
	protected String nombre;
	// Ubicación para escribir los mensajes recibidos, relacionados con los aportes de la discusión
	TextArea areamensajes;

Métodos:	// Constructor
	public DiscConsumidor(String nombre, TextArea areamsg) {
	this.nombre=nombre;
	this.areamensajes=areamsg;
	}
	
	// dataReceived.
	public synchronized void dataReceived(Data datos) {
	String mensaje;
	int posicion = 0;
	String emisor = datos.getSenderName();
	String datosenviados = datos.getDataAsString();
	
	// Construye el mensaje y lo agrega al final del área de mensajes de la discusión.
	
	mensaje = "<"+emisor + ">: " + datosenviados + "\n";
	posicion = areamensajes.getText().length();
	areamensajes.insert(mensaje, posicion);
	}

Clase:	**LUDConsumidor.java**
Descripción:	Implementa la interface ChannelConsumer para obtener la lista de autores en la discusión. Recibe los mensajes enviados por el canal creado para la lista de autores en la discusión e implementa el método dataReceived(), verificando si se agrega o se retira al autor de la lista.
Paquetes: usados	com.sun.media.jsdt.*; java.awt.*;

Atributos:	// Nombre del consumidor del canal
	protected String nombre;
	// Ubicación para escribir los mensajes recibidos, relacionados con los autores en la sesión
	List autores;

Métodos:	// Constructor
	public LUDConsumidor(String nombre, List auts) {
	this.nombre=nombre;
	this.autores=auts;
	}
	
	// dataReceived.
	public synchronized void dataReceived(Data datos) {
	
	// valida el primer caracter del mensaje. El resto del mensaje es el nombre del autor.
	// Si el primer caracter del mensaje es
	// 0, agrega el autor la lista autores
	// 1, retira el autor de la lista autores
	}

Clase:	**Votacion.java**
Descripción:	Realiza el proceso general de la votación y construye la respectiva ventana (Frame). Implementa las interfaces WindowListener y ActionListener para el control de los eventos que ocurren en la interface. Define el método para crear o conectarse al canal de la votación.
Paquetes usados	com sun media.jsdt.*,java.awt.event.*; java.awt.*;
Atributos:	WSUsuario wsUsuario; VotConsumidor votConsumidor; Channel canal; Data datos; // variables de la interface gráfica (botones, menú, lista de opciones de la votación, área de mensajes, etc)

Métodos	// Constructor
	Votacion(Frame ventana, TextArea idea, WSUsuario wsUsuario) {
	// crear el menú y la interface gráfica
	// agregar los objetos que realizan una acción utilizando "objeto addActionListener"
	// llama el método conectar() (declarado más abajo)
	}
	
	public void actionPerformed(ActionEvent event){
	// verificar el objeto que realiza la acción Dependiendo del objeto:
	// llamar el método escribirmensaje(String mensaje) (declarado más abajo)
	}
	
	private void conectar(String idea){
	// crea un nuevo canal para la votación o se conecta si ya existe. El canal creado se almacena en el objeto canal
	// crea y agrega el objeto consumidores de las clases VotConsumidor
	}
	
	private void escribirmensaje(String msg) {
	// Escribe el mensaje voto emitido en el canal canal. Dependiendo el voto, escribe 1, 2 o 3 al inicio del mensaje
	}
	
	private void guardavotacion() {
	// Escribe el resultado de la votación en un archivo de texto
	}

Clase:	**VotConsumidor.java**
Descripción:	Implementa la interface ChannelConsumer para obtener el resultado de cada votación. Recibe los mensajes enviados por el canal de la votación e implementa el método dataReceived, en el cual se contabilizan los votos recibidos como mensajes. Por cada canal que se crea para cada votación, se crea un objeto de la clase VotConsumidor.
Paquetes: usados	com.sun.media.jsdt.*; java.awt.*;

Atributos:	// Nombre del consumidor del canal
	protected String nombre;
	// Ubicación para escribir los mensajes recibidos, relacionados con los votos emitidos
	List votos;

Métodos:	// Constructor
	public VotConsumidor(String nombre, List votos) {
	this.nombre=nombre;
	this.votos=votos,
	}
	
	// dataReceived.
	public synchronized void dataReceived(Data datos) {
	String voto;
	int total = 0;
	String datosenviados = datos.getDataAsString();
	
	// si datos enviados = "1", contabiliza voto a favor.
	// si datos enviados = "2", contabiliza voto en contra.
	// si datos enviados = "3", contabiliza otro
	}

Clase:	**EVDialogo.java**
Descripción:	Construye la ventana (Frame) para desplegar los mensajes difundidos a uno o a todos los autores en la sesión. Implementa las interfaces WindowListener y ActionListener para el control de los eventos que ocurren en la ventana.
Paquetes: usados	java.awt.*; java.awt.event.*;
Atributos:	WSUsuario wsUsuario; String mensaje; // variables de la interface gráfica (botones, labels para los mensajes)

Métodos:	// Constructor
	EVDialogo(Frame ventana, String mensaje, String emisor, WSUsuario wsUsuario) {
	// crear la ventana de diálogo, desplegar el mensaje apropiado y el botón "Aceptar" en caso de que sea un
	// mensaje, o los botones "Si" y "No", en caso de que sea una invitación a la discusión
	// agregar los objetos botón utilizando "objeto.addActionListener"
	}
	
	public void actionPerformed(ActionEvent event){
	// verificar si se acepta participar en la discusión y crear el objeto de la clase Discusion
	}

Clase:	**MSGDialogo.java**
Descripción:	Construye la ventana (Frame) que solicita el mensaje para ser enviado a uno o a todos los demás autores. Implementa las interfaces WindowListener y ActionListener para el control de los eventos que ocurren en la ventana. Crea la lista de autores en la sesión y le agrega la opción "todos" como otra opción para el envío del mensaje
Paquetes: usados	java.awt.*; java awt.event.*;
Atributos.	WSUsuario wsUsuario, String mensaje, destino; WSUsuarioFrame wsUsuarioFrame, // variables de la interface gráfica (botón, lista de autores y area del mensaje)

Métodos:	// Constructor
	MSGDialogo(Frame ventana, WSUsuario wsUsuario, WSUsuarioFrame wsUsuarioFrame) {
	// crear la ventana de diálogo
	// llamar el método listaautores() (declarado más abajo)
	// agregar el objeto botón utilizando "objeto addActionListener"
	}
	
	public void listaautores(){
	// crea la lista de autores, obteniéndola del objeto wsUsuario listausuarios. Agrega la opción "todos" a la lista.
	}
	
	public void actionPerformed(ActionEvent event){
	// asigna a la variable destino el valor seleccionado de la lista de autores
	
	// si destino = "todos"
	// - difunde el envío del mensaje para todos
	mensaje="3"+mensaje
	wsUsuarioFrame.escribirevento(mensaje,3,"");
	
	// sino
	// - difunde el envío del mensaje para el autor seleccionado
	mensaje="2"+mensaje
	wsUsuarioFrame.escribirevento(mensaje,2,destino);
	}

Clase:	**WSDialogo.java**
Descripción:	Construye la ventana (Frame) que solicita el tema para generar la discusión o la idea para promover la votación. Implementa las interfaces WindowListener y ActionListener para el control de los eventos que ocurren en la ventana. Crea las clases Discusión o Votación y difunde el evento al canal wsUsuario.canal (registro de actividades de la sesión), al canal wsUsuario.canal_eventos (invitación a discusión o votación promovida) y al canal canal_ldiscusiones (lista de discusiones en proceso).
Paquetes: usados	java awt.*; java.awt event *; com.sun.media.jsdt.*;
Atributos:	WSUsuario wsUsuario; Session sesion; String mensaje, tema, idea; WSCliente cliente, WSUsuarioFrame wsUsuarioFrame; // variables de la interface gráfica (botón y área del mensaje que almacena el tema o la idea solicitada)

Métodos	// Constructor
	WSDialogo(Frame ventana, WSUsuario wsUsuario, WSUsuarioFrame wsUsuarioFrame) {
	// crear la ventana de diálogo con los mensajes apropiados según sea promover votación o generar discusión
	// agregar el objeto botón utilizando "objeto.addActionListener"
	}
	
	public void actionPerformed(ActionEvent event){
	// valida si es discusión generada o votación promovida.
	
	// Si es discusión:
	// - crea el objeto Discusion
	// - difunde la actividad de discusión generada en el registro de actividades de la sesión
	// - difunde la invitación a la discusión
	
	// Si es votación:
	// - crea el objeto Votacion
	// - difunde la actividad de votación promovida en el registro de actividades de la sesión
	// - difunde la votación
	}

Clase:	**WSUsuarioFrame.java**
Descripción:	Construye la ventana (Frame) principal del espacio de trabajo compartido. Implementa las interfaces WindowListener y ActionListener para el control de los eventos que ocurren en la interface. Define el método para conectarse al canal de los eventos de la sesión.
Paquetes: usados	com.sun.media.jsdt.*;java.awt.event *, java awt.*;
Atributos:	WSUsuario wsUsuario; Channel canal_eventos; EVConsumidor evConsumidor; Data datos; // variables de la interface gráfica (botones, menú principal, listas, área de mensajes, etc.)

Métodos:	// Constructor
	WSUsuarioFrame(WSUsuario wsUsuario) {
	this.wsUsuario=wsUsuario;
	// crear el menú principal y la interface gráfica
	// agregar los objetos que realizan una acción utilizando "objeto.addActionListener"
	}
	public void actionPerformed(ActionEvent event){
	// verificar el objeto que realiza la acción. Dependiendo del objeto.
	// - llamar el método conectar() del objeto wsUsuario
	// - crear el objeto WSDialogo, solicitando el tema para generar la discusión o la idea para promover la votación
	// - crear el objeto MSGDialogo para solicitar el mensaje a ser enviado a uno o a los demás autores de la sesión
	// - llamar los métodos definidos más abajo
	}
	private void lista_autores() {
	// crea y despliega la lista de autores en la sesión, consultando los consumidores del canal canal_lusuarios
	}
	private void lista_discusiones{
	// crea y despliega la lista de discusiones, consultando todos los canales de la sesión que tiene el prefijo "D"
	}
	private void escribirevento(String mensaje, int tipo, String usuariodestino) {
	// realiza la difusión de los eventos mediante el canal canal_eventos
	}
	private void escribirmensaje(String mensaje, Channel canal) {
	// Escribe el mensaje en el canal que se pasa como parámetro
	}
	private void conectareventos(){
	// conecta el wsUsuario.cliente al canal EVCanal, creando el objeto evConsumidor (clase EVConsumidor)
	}
	private void conectardiscusion() {
	// crear y desplegar el objeto de la clase Discusion
	}

Apéndice 2. Consideraciones del API JSDT

Previamente a la programación de las clases del espacio de trabajo compartido, es necesario identificar los métodos que se adoptan directamente de las clases e interfaces del JSDT. Estos métodos realizan las funciones para el soporte de la comunicación, sobre el cual se basa la arquitectura del espacio de trabajo.

Inicialmente, es necesario establecer la forma como se almacena toda la información referente a la sesión (clientes, canales, consumidores, etc.). Para ello, el JSDT proporciona la clase Registry, la cual se encarga de crear en memoria el registro de los objetos de la sesión, con sus respectivos nombres, para que estén disponibles para cualquier objeto de la sesión. Para crear el objeto Registry, se emplea la clase RegistryFactory, estableciendo el tipo de implementación que se utiliza en la sesión.

El Registry se inicia en el servidor de la sesión mediante el método startRegistry(tipo) de la clase RegistryFactory. El parámetro tipo, especifica el tipo de implementación utilizado, por ejemplo: RegistryFactory.startRegistry(socket), inicia el Registry utilizando sockets de TCP/IP como protocolo de comunicación.

Una vez se inicia el almacenamiento de la información de la sesión, se realiza la creación de la sesión de grupo. Para ello, se utiliza el método createSession(cliente, url, autojoin) de la clase SessionFactory. Los tres parámetros de este método indican el cliente que crea la sesión, el url de la sesión y si dicho cliente se conecta (*"autojoin"*) o no a la sesión, al momento de crearla. La siguiente secuencia, define la sintaxis para la creación de la sesión:

```
URLString url;        // define el tipo especial del url de la sesión
Cliente cliente;      // define el objeto de cliente la clase Cliente
Session sesion;       // define el objeto de la clase Session.

url=URLString.createSessionURL("mipc.midominio.com",4466,"socket","WSSesion");
cliente = new Cliente("Servidor");
sesion= SessionFactory.createSession(cliente, url, false);
```

En esta secuencia, se especifica que el cliente llamado "Servidor" no se conecta a la sesión en el momento en que la crea, porque como se ha explicado, el servidor no utiliza la información que se comparte en la sesión.

Para que un cliente se una a la sesión de grupo creada, realiza la llamada al método createSession con una sintaxis similar a la anterior. Debido a que la sesión definida por el url ya existe, este método devuelve la referencia al objeto sesión creado en el servidor. Por ejemplo, en la secuencia:

```
cliente = new Cliente("Autor 1");
sesion= SessionFactory.createSession(cliente, url, true);
```

el cliente que identifica al Autor 1 es agregado a la sesión que ya existe y la referencia de dicha sesión se almacena en el objeto sesion del Autor 1.

Cuando ha creado el objeto sesión, el cliente puede crear los diversos canales para compartir la información. Esto se lleva a cabo mediante el método createChannel(cliente, nombre, confiable, ordenado, autojoin) del objeto sesion (clase Session). Los parámetros de este método indican el cliente que crea el canal, el nombre del canal, si el canal es de tipo confiable, si el canal es ordenado (los mensajes se entregan en el orden en que son escritos en el canal) y si el cliente se conecta o no al canal, en el momento en que lo crea. La siguiente secuencia define la sintaxis para la creación del canal:

```
sesion.createChannel(cliente,"WSCanal", true, true, false);
```

En esta secuencia el método createChannel crea el canal "WSCanal" indicando que es confiable y ordenado, y que el cliente que lo crea no se conecta a él en el momento de la creación. Para que un cliente se agregue a un canal que ya existe, realiza la llamada al método createChannel, con una sintaxis similar a la de la secuencia anterior. Por ejemplo la secuencia:

```
Channel canal;                    // define el objeto de la clase Channel
cliente = new Cliente("Autor 1");
canal= sesion.createChannel(cliente,"WSCanal", true, true, true);
```

agrega el objeto cliente que identifica al Autor 1, al canal WSCanal existente. La referencia de este canal se almacena en el objeto canal del Autor 1.

Los clientes que se han agregado a un canal pueden consumir los mensajes que se envían a través de él. Para ello, cada cliente crea un objeto de la clase consumidor (clase que se establece para la interface com.sun.media.jsdt.ChannelConsumer) y se agrega mediante el método addConsumer(cliente, consumidor) del objeto de la clase Channel. Por ejemplo la secuencia:

```
cliente = new Cliente("Autor 1");
consumidor= new Consumidor(cliente. nombre, String mensajes[]);
canal.addConsumer(cliente,consumidor);
```

asigna el cliente que identifica al Autor 1 como consumidor de canal. El método constructor Consumidor(cliente. nombre, String mensajes[]), es sólo un ejemplo que indica que los mensajes recibidos serán manipulados como un arreglo de cadenas de caracteres.

Finalmente, para que un cliente envíe un mensaje a todos los clientes consumidores de un canal, el mensaje debe codificarse en un objeto Java Data y enviarse mediante el método sendToAll(cliente, Data) de la clase Channel. Por ejemplo la secuencia:

```
datos = new Data(String mensaje);
datos.setPriority(Channel.HIGH_PRIORITY);
canal.sendToAll(cliente, datos);
```

envía un mensaje del emisor cliente, codificado en el objeto datos, a todos los clientes consumidores de canal. Dicho mensaje se envía con una prioridad alta, para que se entregue primero que los otros mensajes de menor prioridad.

La clase Channel del JSDT, también define los métodos sendToOthers(cliente,Data) y sendToClient(cliente_emisor,cliente.nombre_receptor, Data), para enviar mensajes a todos los consumidores diferentes al emisor y a un consumidor específico, respectivamente.