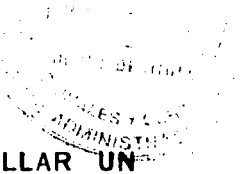
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

"ACATLAN"



"TECNICA PARA DESARROLLAR UN ADMINISTRADOR DE VERSIONES DENTRO DE UN SISTEMA DE EDICION COLABORATIVA"

T E S I N A

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A

JESUS EDGAR GUZMAN CAN

Asesor: Anabel Moreno Baltazar

Naucalpan Edo. de México Mayo 2003

A G R A D E C I M I E N T O S

A mis padres Nice Guadalupe Can Echeverría y Ángel Guzmán Ortiz quienes me han apoyado en todo lo que haga, sin mirar las ventajas o desventajas de mis decisiones. Por las largas pláticas que hemos tenido, sus consejos, regaños, por esto y más, les agradezco su amor y apoyo incondicional que nos tienen.

A mis hermanos, Guadalupe Edith (futura Comadre) y Pedro Ángel (Compadre) por su apoyo en las buenas y en las malas, además agradezco sus consejos que me dan. Aunque están juntos y nos les gusto no me importa los quiero igual a los dos.

A mi cuñado Sergio por su ayuda a tomar decisiones y que siempre esta conmigo para escuchar, y aconsejarme. A mi cuñada Norma por sus consejos en algunas decisiones que he tomado con la ayuda de mi hermano.

A mis sobrinas Linette y María Fernanda por estar en este momento.

A la banderola de la ENEP ACATLAN que siempre son y serán mis amigos, por su apoyo para poder lograr esta meta en mi vida. En especial a Lupita (R) y Araceli.

A mi Asesora Anabel por su apoyo y asesoría para poder concluir este trabajo de titulación.

A mis compañeros del IPN que me han apoyado, aconsejado y regañado para poder cumplir con esta meta de mi vida.

Principalmente a Dios por ser más que una fe y una religión.

G R A C I A S

Í N D I C E

I N T R O D U C C I Ó N

El trabajo maneja aspectos de la carrera de Matemáticas Aplicadas y Computación, los cuales se retomarán como sistemas operativos y teleprocesos. Propone una técnica para la creación de un administrador de versiones.

El trabajo de investigación es bibliográfico, dando información acerca de que es el control de versiones, la edición colaborativa y una propuesta de la técnica para el diseño de un administrador de versiones para un sistema de edición colaborativa. Está compuesta de cuatro capítulos, los cuales se presentan brevemente:

Capítulo 1, se desarrolla un marco de investigación indicando en que posición se encuentra el tema de investigación dentro de los sistemas distribuidos, partiendo de lo general a lo particular, abarcando Sistemas Distribuidos, CSCW, Groupware, Edición Colaborativa y Control de Versiones.

En el capítulo 2 se recopila información de algunos modelos realizados por investigadores para poder atacar el problema del control de versiones en un ambiente de edición colaborativa. Los modelos investigados fueron: Palimpsest, Chimera, Contexto Anidados y WebDAV.

En el capítulo 3 se expone acerca de los sistemas que se han realizado que incorporan un administrador de versiones, en un ambiente colaborativo, los sistemas que se estudiaron son: MUCH, HyperProp, ClearCase y CVS. De los cuales únicamente el CVS no es un prototipo.

En el capítulo 4 se expone que es la edición colaborativa y como esta relacionada con las dinámicas de grupo y la técnica para el desarrollo de un administrador de versiones y el manejo de las mismas.

CAPÍTULO 1

ANTECEDENTES

En el capítulo 1, se hablará de la posición en que se encuentra el control de versiones dentro del CSCW y la edición colaborativa. Se partirá de lo general para llegar a lo particular, comenzando con los sistemas distribuidos y terminando con el control de versiones.

1.1 Antecedentes

Los Sistemas Distribuidos tienen sus antecedentes en los Sistemas Operativos que surgen en la década de los 40 y 50, cuando las computadoras eran únicamente mono-procesadores, es decir, que las máquinas podían soportar solamente un usuario a la vez. Por tal motivo, los sistemas operativos volvieron a su principal propósito: manejar y administrar los recursos (software, hardware y datos) de las máquinas.

Una definición de un sistema operativo es la siguiente:

> "Es el que controla todos los recursos de la computadora y ofrece la base sobre la cual pueden escribirse los programas de aplicación."[Tanenbaum,1]

Lo anterior muestra, claramente que un sistema operativo es el que administra, gestiona y controla todos los recursos de la máquina (software, hardware y datos).

Con la imposibilidad que los sistemas operativos permitieran compartir recursos y que sólo posibilitarán el trabajo a un usuario a la vez, se empezó a diseñar un sistema que pudiera realizar la compartición de recursos y al mismo tiempo permitiera el acceso a varias personas a trabajar.

1.2 Sistemas Distribuidos

Considerando los avances en los Sistemas Operativos tradicionales como el MS-DOS y Windows se llegó a la construcción de los Sistemas Distribuidos. Los Sistemas Distribuidos permiten la ejecución de muchos procesos, la administración y el control de los recursos y múltiples dispositivos de almacenamiento conectados a través de una red.

Dada la importancia de estos sistemas se hace necesaria la inclusión de las siguientes definiciones:

1) "Un Sistema Distribuido es una colección de computadoras independientes que aparecen ante los usuarios del sistema como una única computadora"[Tanenbaum,2]

2) " Un Sistema Distribuido consiste de una colección de computadoras autónomas ligadas por una red y están equipadas con un software de un sistema distribuido" [Coulouris&Jean,88]

3) "Un sistema distribuido es uno que mira a sus usuarios como un sistema activo centralizado ordinario, pero se ejecuta sobre múltiples computadoras independientes. El concepto clave aquí es la transparencia, en otras palabras, el uso de múltiples procesadores debe ser invisible al usuario. El usuario ve al sistema como un "único-procesador virtual", no como un conjunto de máquinas distintas."[Tanenbaum&Van Renesse,85]

Es importante destacar las características importantes de este sistema tales como: [Coulouris&Jean,88]:

1- **Soporta el manejo de los recursos compartidos:** Los recursos que comparte un sistema distribuido son componentes de Hardware y Software. Lo que se comparte es:

   a) **Hardware:** Las computadoras pueden compartir impresoras, discos duros y otros periféricos y así reducir costos.

   b) **Software:** Como los desarrolladores de software deben trabajar en equipo, puedan compartir las mismas herramientas de desarrollo, compiladores, bibliotecas, editores, etc. Y cuando se crea una herramienta, se comparte a los demás desarrolladores cuando estos la soliciten.

   c) **Datos:** El poder accesar a las bases de datos de las demás máquinas para así manipular la información deseada.

Debido al gran desarrollo de aplicaciones para redes y los Sistemas Distribuidos como soporte de grupos que trabajan en forma colaborativa en proyectos, lideres de proyectos, en la enseñanza revistas y otras metas que se logran en conjunto. De ahí surge el ***CSCW*** (Trabajo Cooperativo Asistido por la Computadora)[1], que depende altamente de la compartición de los datos entre programas, corriendo en diferentes estaciones de trabajo.

En la compartición de los recursos se tienen dos modelos que son utilizados por los Sistemas Distribuidos [Coulouris&Jean,88]:

1) **Modelo Cliente – Servidor:** Este modelo es un conjunto de procesos del servidor, cada acción lo toma como un recurso de un mismo tipo y colección de procesos del cliente. En este modelo, todos los recursos compartidos son controlados y manejados por un proceso del servidor. Cuando un cliente necesita un recurso lo que hace es mandar una petición al servidor y éste se encargará de proporcionarle el recurso que se solicitó, siempre y cuando el servidor reconozca al cliente que solicita el recurso. Este modelo provee un aprovechamiento eficaz y efectivo del objetivo general para la compartición de información y recursos en un sistema

[1] Del término en Inglés "Computer Supported Cooperative Work"

distribuido, el modelo puede ser implementado en una variedad de diferentes plataformas y hardware.

Las computadoras pueden ser clientes y servidores a la vez, un proceso del servidor puede usar el servicio de otro servidor, apareciendo como cliente después. El modelo es bueno en los sistemas actuales donde se manejan diferentes tipos de recursos compartidos como: correo electrónico, almacenamiento de disco, archivos, redes grandes sincronizadas por un reloj. Siempre recordando que no todo se puede compartir, tal es el caso de la memoria RAM, el procesador central y la interfaz local de red usualmente vistos como un conjunto mínimo de lo que no se puede compartir.

2) **Modelo basado en objetos:** En el modelo para los Sistemas Distribuidos, cada recurso compartido es tomado como un objeto. Los objetos son identificados de manera única y permanente. Cuando se quiera utilizar un recurso compartido, el programa manda una petición del objeto que se requiere. Este modelo es atractivo por ser simple y flexible, habilita que todos los recursos puedan ser vistos de manera uniforme por los usuarios.

   En el modelo, los objetos pueden actuar como recursos del usuario y como recursos del manejador (el manejador es el que se encarga de hacer una colección de procedimientos y datos obtenidos, que juntos caracterizan una clase de objetos).

Los recursos de los Sistemas Distribuidos son físicamente encapsulados dentro de una de las computadoras y pueden únicamente accesarlo desde otra computadora por medio de la comunicación.

2- **Extensibilidad:** La extensibilidad de un sistema distribuido es la característica que determina cuándo el sistema puede ser extendido de varias maneras, desde el hardware y software. Un sistema puede ser abierto o cerrado con respecto al hardware o con respecto al software. La extensibilidad de Sistemas Distribuidos está determinado por el grado por el cual un recurso compartido puede ser adicionado sin una perturbación, cambio o duplicación del sistema existente. La extensibilidad se logra especificando y documentando el identificador que tiene el software. Los sistemas que son creados para asistir a la compartición de los recursos son llamados Sistemas Distribuidos abiertos. Los Sistemas Distribuidos pueden también ser extendidos a nivel del hardware añadiendo computadoras a la red y a nivel de software introduciendo nuevos servicios y capacidad de compartir los recursos de los programas de aplicación. Un ejemplo de un sistema abierto es el sistema operativo UNIX, el cual esta programado en lenguaje C, y permite a los programadores acceder a todos los recursos fácilmente a través de un conjunto de procedimientos que se denominan "llamadas al sistema"[2].

3- **Concurrencia:** Es cuando varios procesos se ejecutan al mismo tiempo en una máquina, con un sólo procesador, esto implica planificación del CPU. Cuando son varios procesadores se le llama paralelismo. En un sistema distribuido que se basa en el

[2] Del término en Inglés "System Calls".

modelo de recursos compartidos, los procesos pueden correr paralelamente por dos razones:

a) Si muchos usuarios simultáneamente invocan programas de aplicación, se origina más de un llamado en la máquina. Si la computadora tiene únicamente un solo procesador y corre más de una aplicación, los procesos se ejecutan en un modo de compartición de memoria..

b) Muchos procesos del servidor corren concurrentemente, cada proceso del servidor responde a diferentes solicitudes desde los procesos del cliente. Se originan solicitudes desde la existencia de uno o más procesos del servidor para cada tipo de recurso. Estos procesos normalmente corre en equipos adicionales, posibilitando cada proceso del servidor que corra o proceda en forma paralela.

4- **Escalabilidad:** Capacidad de un sistema para operar de manera efectiva independientemente de su tamaño. La existencia de crecimiento en el sistema no implica modificaciones de la arquitectura básica del mismo. Los Sistemas Distribuidos deben operar eficientemente a pesar de que pueden crecer en sus componentes. El sistema distribuido más pequeño es el que consiste de una estación de trabajo y un servidor, considerando que se puede construir un sistema distribuido alrededor de una simple red local que podría contener cientos de estaciones de trabajo y muchos servidores, servidores de impresora y otros tipos de servidores.

5- **Transparencia:** Se define en la forma en que el usuario percibe al sistema como un todo, más que como una colección de componentes independientes. Existen varios tipos de transparencias.

Transparencia de:

a) **Acceso:** Permite a los objetos de informaciones distantes y locales ser accesados usando operaciones idénticas.

b) **Lugar:** Permite a los objetos de información ser accesados sin conocimiento de su ubicación.

c) **Concurrencia:** Permite el trabajo de modo concurrente sin que se afecten otros procesos.

d) **Replicación:** Permite múltiples instantes de los objetos de información ser usados en ejecución sin conocimiento de las replicas por usuarios o por programas de aplicación.

e) **Fallo:** Permite el ocultamiento de fallas, permitiendo a usuarios y programas de aplicación completar sus tareas a pesar de las fallas en hardware o componentes del software.

f) **Migración:** Permite mover los objetos de información dentro de un sistema sin afectar las operaciones de los usuarios o programas de aplicación.

g) **Desempeño:** Permite a los sistemas ser reconfigurados para un mejoramiento del desempeño con cargas variadas.

h) **Escalabilidad:** Permite al sistema y aplicaciones expandirse en escala sin cambios en la estructura del sistema o en el algoritmo de la aplicación.

Las formas de transparencias más importantes son la de *acceso* y la de *lugar* que se utilizan en la transparencia de red.

6) **Tolerancia a fallas:** Los Sistemas Distribuidos utilizan la concurrencia. El más importante aspecto en los Sistemas Distribuidos es la posibilidad de particionar el sistema: una parte del sistema podría fallar y recuperarse, mientras el resto del sistema sigue trabajando. Este tienen un impacto significativo en el diseño de los Sistemas Distribuidos. Los Sistemas Distribuidos introducen la posibilidad de fallos independientes – componentes que fallan individualmente mientras otros continúan corriendo – y también tienen mucho tiempo de espera para la comunicación entre multiprocesadores. Estos atributos de los Sistemas Distribuidos no requieren diferentes técnicas para especificaciones y verificaciones. Sin embargo, el deseo de tolerar las fallas y las demoras de comunicación, algunas veces han forzado al diseñador de sistema a proveer de aplicaciones más robustas a los clientes. El diseño de la tolerancia a falla en las computadoras responden a dos propuestas que son:

   - Redundancia del hardware.

   - Recobrar el software o la información.

Para la redundancia del hardware se utiliza la arquitectura en espera - listo (HOT STANBY). Para recobrar la información se utiliza la técnica del desplazamiento (ROLL-BACK) que es la recuperación de un dato permanente.

Algunos diseños de Sistemas Distribuidos son asíncronos (componente que opera con su propia velocidad y pueden esperar hasta que sean requeridos). La velocidad está determinada por la velocidad de los componentes compartidos.

Algunas aplicaciones de los Sistemas Distribuidos son [Coulouris&Jean,88]:

- **Comerciales:** Muchos comercios que se encuentran físicamente distribuidos utilizan el procesamiento de datos y sistemas de información. Ejemplos las aerolíneas utilizan los sistemas para poder reservar y vender boletos. Las redes que operan en los bancos que asisten para optimizar los trabajos que se tienen de transacciones y verificaciones de cuentas. Estos requieren de aplicaciones que sean seguras contra una interferencia externa y privacidad de información.

- **Redes grandes:** Las redes grandes han incrementado su popularidad permitiendo la interconexión de computadoras. Internet es ejemplo de las redes grandes, y una aplicación sencilla es el correo electrónico, este es una simple aplicación distribuida debido a que la gente que escribe puede encontrarse en lugares distantes.

- **Multimedia y conferencias:** Las aplicaciones multimedia utilizan las representaciones digitales de imágenes fotográficas, audio y secuencias de video para ayudar al aprendizaje por medio de computadoras, para videoconferencias y trabajo cooperativo, para jugar y comprar.

- **CSCW[3]:** Los Sistemas Distribuidos son utilizados en esta área para poder asistir al trabajo colaborativo. En este punto profundizaremos, ya que forma parte de este trabajo de investigación.

1.3 CSCW

El término CSCW significa *Trabajo Colaborativo Asistido por Computadora.* Este es un concepto usado en muchas áreas académicas y en los círculos de investigación que utilizan computadora y telecomunicaciones (es el campo que estudia el manejo de la información por vía electrónica a largas distancias), por un grupo de personas que trabajan conjuntamente.

Para CSCW es importante el estudio de la actividad humana para poder trabajar juntos ayudándose de la tecnología en los procesos de trabajo como: el intercambio de la información, la compartición de la misma, la autoría conjunta de textos, la toma de decisiones, etc.

CSCW estudia la forma en que la gente trabaja en grupos y cómo la computadora interacciona con los grupos, se le conoce como "Conciencia de grupo".

El CSCW está muy relacionado con disciplinas de estudio de la conducta humana, también sobre el manejo del lenguaje y las computadoras. Una de las disciplinas que tiene mayor relevancia es la psicología social. Es una parte importante en el campo de la colaboración debido a que examina el impacto que tiene la tecnología sobre los grupos sociales y el comportamiento para ser considerado en el diseño de nueva tecnología.

También se encarga de estudiar cómo la gente interactúa en un grupo y cómo los individuos pueden afectar el comportamiento de una persona y el trabajo mismo.

Aparte de la psicología social, también se apoya de la Etnografía que es la ciencia que tiene como objeto el estudio de las razas y los pueblos desde un punto de vista analítico y descriptivo.

[3] Del término en Inglés "Computer Supported Cooperative Work"

Otro significado de etnografía [Twidale&Nichols,98], se involucra en el estudio de cómo la gente hace su trabajo en la vida real tomando en cuenta lo que ocurre de acuerdo a sus acciones, además intentando comprender que la gente trabaja de acuerdo a un contexto propio y que no sale del mismo y para describir lo que la gente realmente hace con su trabajo. La etnografía juega un papel muy importante en CSCW en un aspecto sociológico.

El objetivo del CSCW es la interacción social de la gente y el apoyo que se tenga para poder trabajar con la tecnología. La meta de CSCW es la de descubrir maneras de usar la tecnología de las computadoras para fomentar y mejorar el trabajo de grupos, mediante el apoyo de la tecnología, el tiempo y los grupos sociales.

Para satisfacer las metas del CSCW se ha desarrollado tecnología, y métodos como son [Pfeifer et al,95]:

WYSIWIS

El WYSIWIS en Inglés significa *"What You See is What I See"* que se traduce como el método Lo Qué Tú Ves es Lo Qué Yo Veo. Es analógico a dos personas de cada hogar que ven la televisión, en el sentido de que tú ves lo que yo quiero ver cuando sólo se tiene un televisor, situación que, sucede muy a menudo por ser una acción normal de nuestra vida cotidiana. Este concepto se extendió hacia las computadoras y permitió que la gente que trabajaba en grupos pudiera entender el concepto y así manejarlo, recordando que el método puede ser no con dos personas sino con un grupo grande de personas que están trabajando colaborativamente.

INTEGRACIÓN A TRAVÉS DE TAREAS DIFERENTES

Cuando se integran todos los componentes de diferentes tareas o campos de acciones, se reducen los problemas entre los componentes de aplicaciones diferentes y se provoca que las herramientas que se utilizan sean más grandes y más flexibles.

GESTIÓN DE TIEMPO

Hay productos de CSCW que se ocupan para gestionar el tiempo de una empresa, sacar calendarios de procesos y proyectar estas planillas que ayuden a los gerentes a llevar una mejor gestión de tiempo de los empleados y un calendario óptimo del trabajo, como también a los subordinados sepan administrar su tiempo mejor.

MULTIMEDIA

CSCW puede aprovechar las gráficas, presentaciones, video, audio para proveer una interfaz más natural, además de ser amigable con el usuario o usuarios que estén laborando en un trabajo por equipo.

TERMINACIÓN DE UN PROGRAMA DE USUARIO

Las herramientas CSCW deben ser lo suficiente flexible con el usuario para que él pueda cambiar o modificar las herramientas y así favorecer las necesidades requeridas en ese instante por el usuario. También el usuario debería ser capaz de cambiar la herramienta dependiendo del tipo de interacción colaborativa que esté ocupando.

Estas son algunas tecnologías y conceptos que contribuyen a satisfacer las metas del CSCW.

Una de las herramientas más recientes es la tecnología REALIDAD VIRTUAL que nos hace ver a la gente más cerca, aunque se encuentre distribuida geográficamente en el mundo, logrando que exista una interacción más cercana con la gente por simular que se encuentran en el mismo lugar.

Existen varias clasificaciones de los tipos de sistemas de CSCW determinada por el lugar y el tiempo de la interacción colaborativa que están siendo asistidas. Esta clasificación se propuso en 1988. La colaboración puede ser entre personas que se encuentran en diferentes lugares o en un mismo lugar, también puede ocurrir en un mismo tiempo (sincrono) o en tiempos diferentes (asincrono).

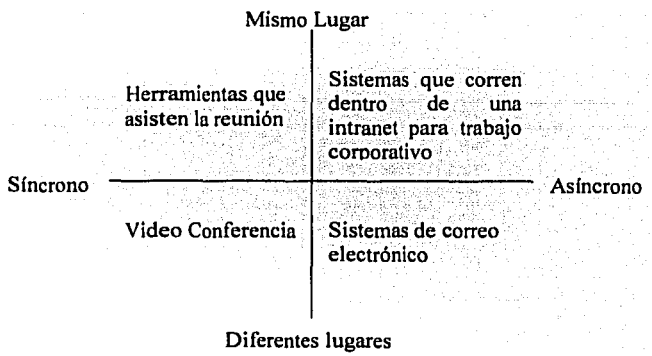


Figura 1 Clasificación en tiempo y lugar

En la figura 1[Pfeifer et al. 95] se muestra la clasificación de la colaboración por dos elementos principales que son: tiempo y lugar. Mostrando cómo están clasificadas las herramientas utilizadas por el CSCW.

1.3.1 TIPOS DE SISTEMAS CSCW

Existen diferentes tipos de sistemas [Twidale&Nichols,98] que han sido desarrollados para la investigación de cómo se asiste a los diferentes tipos de interacción colaborativa. Muchos sistemas son creados en laboratorios, pero al salir al mercado en un trabajo real, se aprecia que no funcionan para lo que fueron creados. Sólo algunos sistemas han tenido éxito dentro de la tecnología colaborativa, tales como la Web, e-mail y video conferencias.

En esta sección se hablara de algunos sistemas CSCW como:

a) **Asistencia para reuniones o videoconferencia:** En las empresas se tiene un gran número de reuniones que pueden ser en el mismo lugar, frente a frente o mismo tiempo pero con personas que se encuentran en diferentes lugares. Esto significa una gran perdida de tiempo y dinero. La tecnología actual podría enfocar su asistencia para las reuniones a larga distancia. Esto se realiza a través de un teclado y un monitor que se encuentran en su escritorio donde podrá observar a las otras personas de la reunión. Antes se podía utilizar un pizarrón blanco, mapas y transparencias que no permitían gran flexibilidad de tiempo y velocidad en las reuniones, ahora se utilizan presentaciones computaciones como diapositivas, diagramas, mapas conceptuales con esto se facilita el manejo de grandes cantidades de información.

   Una adición a esta tecnología es la asistencia de reuniones distribuidas, sistemas que incluyen video en vivo y conjunto de equipo para visualizar. El uso de micrófonos, control de piso, esquema a distancia y un apuntador puede ser un poco complicado cuando se asiste una reunión distribuida, pero el resultado obtenido es más en tiempo real. Son conocidas las videoconferencias, sistemas para la asistencia de las reuniones que se hacen en un lugar pero las personas no se encuentran físicamente ahí, o también se ocupa para las conferencias que se hacen en una Universidad y se transmite en tiempo real a distintas partes del mundo. Esto ha crecido gracias a Internet debido que es un canal de comunicación muy solicitado para las videoconferencias.

   Mucho de lo que se investiga en CSCW gira en torno a la colaboración en diferente lugar pero con tiempo sincrono. En particular, los investigadores tratan de entender la importancia de las videoconferencias. Se sabe que las videoconferencias son caras, además de que es importante conocer bastante el uso de esta tecnología para poder tomar ventaja sobre el costo. En las investigaciones que se han realizado sobre pruebas de las videoconferencias se muestra que importa más el audio que el video. Su finalidad de los sistemas CSCW es ayudar a la gente en su trabajo diario.

b) **Escritura Colaborativa:** Este es nuestro punto de interés, debido a que nosotros nos encargaremos de explorar lo que es la edición colaborativa. Se han desarrollado muchos proyectos que tienen que ver con el uso de la edición colaborativa o cooperativa con varias personas que se encuentran en diferentes partes del mundo. El proceso de la escritura colaborativa de un documento es claramente la mayor

actividad de los investigadores, aplicado en la ciencia, pero también puede estar en el comercio.

Un artículo puede ser escrito colectivamente desde el principio o pasar a través de una serie de revisiones y ediciones. La asistencia podría ser una escritura colaborativa sincronizada a distancia, donde los autores discuten y revisan un documento como si ellos estuvieran en el mismo lugar, aunque se encuentren a miles de kilómetros entre sí y en un mismo tiempo. Esta es una gran ventaja de la nueva tecnología contra la edición tradicional. La edición tradicional se llevaba acabo, cuando las personas se encontraban en el mismo lugar pero sólo se podían comunicar con las otras personas para que les dieran su punto de vista del documento por medio del teléfono, esto era una perdida de tiempo y de dinero. Actualmente, se trabaja con la edición colaborativa que nos facilita en mucho el tiempo de edición de un documento. También la edición colaborativa puede ser asíncrona aquí las personas trabajan con el documento en diferente tiempo. Dentro de la edición colaborativa se encuentran los controles de versiones que se dan cuando un documento se está trabajando con varios autores, estos mismos tienen el poder de modificarlo y crear versiones del documento original para que puedan ser vistas por los demás autores relacionados con él.

En los estudios recientes se ha mostrado que aunque la escritura colaborativa es un área de investigación nueva en el CSCW, la tarea es compleja y existe una problemática con el diseño de software. Actualmente el software moderno está ayudando mucho al campo de la edición o escritura colaborativa. Los desarrolladores de este nuevo software indican que las actuales herramientas son buenas para una tarea específica, pero todavía falta para que sean de uso general. [Twidale&Nichols,98]

1.4 GROUPWARE[4]

Los productos comerciales de CSCW son frecuentemente referidos como ejemplos de groupware. Este término es usado algunas veces como sinónimo de tecnología CSCW, otros lo definen como el software enfocado para pequeños grupos, no organizaciones muy grandes que necesitan ser asistidas.

La meta del groupware es asistir a los grupos en comunicaciones colaborativas y coordinar sus actividades, es importante definir primero que es groupware en términos de las metas comunes de un grupo y sus necesidades para compartir su ambiente, así pues se define como:

"Sistemas basados en computadoras que asisten a grupos de personas comprometidas en tareas comunes o metas y que provea una interfaz en un ambiente compartido" [Ellis et all, 93].

Otra definición de Groupware es [Lynch et al, 90]:

[4] Nombre tomado para especificar los sistemas que soportan el trabajo colaborativo.

- El groupware hace que el usuario sepa que es parte de un grupo, mientras que otros tipos de software se preocupan por proteger y esconder a los usuarios uno de otro.

- El groupware es software que enfatiza el ambiente múltiple del usuario, coordinando y dirigiendo cada parte del ambiente sin crear conflictos entre ella.

Las palabras como tareas comunes y un ambiente compartido son muy importantes en la definición, debido que el software de CSCW debe trabajar con grupos de personas que tengan las mismas metas, además de que el ambiente de trabajo debe de ser compartido para poder así trabajar colaborativamente. Dentro de la definición no se especifica sobre como deben trabajar las personas, esto es, si en momentos simultáneos o en tiempos desfasados.

Groupware puede trabajar con tiempos asincronos y se le conoce como Groupware de tiempo no real o con tiempos sincronos y se le conoce como Groupware de tiempo real.

Dentro del Groupware se encuentra el correo electrónico, sistemas electrónicos de reunión, video conferencias, procesos de reingeniería. Esta tecnología, la cual soporta la colaboración actualmente tiene una gran demanda debido al reconocimiento de los factores que esta tecnología integra y las ventajas que se le puedan sacar a la misma.

Los sistemas groupware parten del paradigma de qué es lo que requiere la gente para poder trabajar juntos. El groupware maximiza la interacción humana mientras minimiza la interfaz tecnológica, es decir, lo que pide el groupware es el apoyo que se deben dar las personas, aquí es donde se integra lo que significa el concepto de conciencia de grupo, importa más el trabajo que se realiza en grupo que la interacción con la tecnología.

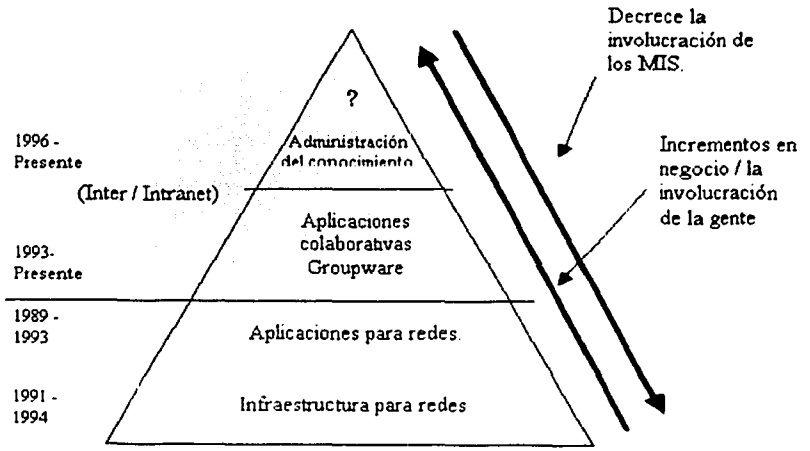


Figura 2 Pirámide del balance de la tecnología y la gente

La figura 2 muestra la pirámide de cómo está el balance sobre la tecnología y la gente. Señala cómo decreció la dependencia de las personas por la tecnología y cómo a crecido el involucramiento de la gente dentro del negocio o de la compañía. Con el paso del tiempo las personas han empezado a pensar cómo involucrarse más al negocio que lugar de que la tecnología se involucre. [Coleman,95]

Existen diferentes tipos de aplicaciones de groupware que son [Bock,93]:

- **Los agentes activos de información:** Este tipo de aplicación nos ayuda a tratar con problemas de excesos de información. El correo electrónico es un problema cuando éste tiene mucha información. Por ello se necesita algo que nosotros podamos usar con una estructura de mensajes para clasificar los correos dentro de ciertas categorías. Los filtros de información son intuitivos y obvios, ellos miran qué pasa en las oficinas cuando nosotros confrontamos un desorden de información. Un ejemplo, son los filtros para el correo electrónico que utiliza algunas instrucciones de programación como IF.... THEN, esto se expresa de la siguiente manera, que cuando llega un correo si no cumple con alguna condición lo que se hace es no pasarlo al buzón.

- **Coordinación de actividades y ligas:** Esta aplicación son los flujos que tiene el trabajo para facilitar la modificación de los agentes de reglas, esta es otra clase de sistema groupware que involucra los vínculos de hipertexto. Esto es, una idea que se empezó a trabajar en la Segunda Guerra Mundial con Vannevar Bus[Bus,45] que observó que *"El ser humano considera operar por medio de la asociación con otros pensamientos y personas. Cuando un artículo es agarrado, este rompe instantáneamente al próximo, que es sugerido por la asociación de sus pensamientos"*[5]. Más bien usando planes de indexación artificial, al buscar la información ésta debería ser más natural como nuestros primeros pensamientos sobre una idea, entonces se puede sacar una asociación para otro. Con el creciente poder de las computadoras de escritorio y las arquitecturas cliente – servidor, nosotros podemos ahora anticipar sistemas que podrían usar la asociación común o vínculos, entre nodos aleatorios de información. Los coordinadores podrían ayudar con los problemas explícitos del manejo de actividades interrelacionadas. Cuando se hace un trabajo complejo y con varias personas se pueden definir diferentes problemas como: identificar a los participantes, programar reuniones, discutir puntos de vista, técnicas de análisis, asignación de tiempos de acción entre otros. Nosotros podemos manejar una serie de requerimientos, acciones y asignaciones. Cada especificación es un nodo de información, que podemos identificar por medio de su información con un proceso de trabajo e identificar vinculaciones entre tareas y coordinar grupos con idénticas actividades.

- **Asistencia a reuniones:** Esta aplicación involucra sistemas con las reuniones electrónicas, las cuales podrían tener participantes en el mismo salón o con algunos individuos en localidades remotas. Seguramente por varios años los usuarios han tenido dispositivos de comunicaciones en sus salones de conferencia tales como:

[5] Vannevar Bus, "As We May Think", Atlantic Monthly, May 1945.

teléfonos, pantallas de proyección y altavoz del presentador o moderador. Sin embargo, los dispositivos de escritorio y los Sistemas Distribuidos proporcionaron nuevas aplicaciones como la posibilidad de asistir las reuniones con mayores enfoques, metas orientadas y de manera interactiva. En las reuniones las muestras o las imágenes que se ven son compartidas con todos los usuarios. Cuando los usuarios trabajan juntos en un problema, todo podría ser visto como una misma información en una computadora personal, como si miráramos un pizarrón, esto es cuando se encuentran los participantes de la reunión en un mismo lugar, pero ¿que pasa cuando la reunión se hace con los integrantes distribuidos en todas partes del mundo?, lo que sucede es que cada uno ve una sola pantalla o un pizarrón. Los sistemas de soporte a reuniones podrían ser usados por integrantes de muchas actividades asociadas con metas en común.

Para tener un buen sistema groupware se deben incorporar filtros de información, sistemas de flujo de trabajo, vínculos de hipertexto y asistencia o soporte de reuniones en tiempo real. Los usuarios podrían interactuar intuitivamente con este ambiente. Estos sistemas fueron inicialmente diseñados por un seguimiento de una simple visión: que los sistemas deberían personificar un sentido de trabajo en equipo y de colaboración, en su seguimiento, estos sistemas prometen tener un efecto con un alcance lejano sobre el diseño de tareas orgánicas, acerca de cómo la gente realmente organiza su trabajo para obtener un máximo beneficio de la tecnología de información que se presenta.

A continuación se presenta una taxonomía de los sistemas de groupware [Ellis et al,93]:

- El Groupware ayuda a grupos cara a cara o a grupos que se encuentran distribuidos en varios lugares. Además un sistema groupware puede ser utilizado para mejorar la comunicación entre las personas y colaborar sin una interacción en tiempo real. El tiempo y el espacio son considerados para las cuatro categorías de groupware que pueden ser representadas por medio de la siguiente matriz de 2 x 2.

	Mismo tiempo	Diferente tiempo
Mismo lugar	Interacción cara a cara	Interacción asincrona
Diferente lugar	Interacción distribuida síncrona	Interacción distribuida asincrona

Figura 3 Matriz de espacio y tiempo de los sistemas Groupware.

La figura 3 muestra como se realizan las interacciones de espacio y tiempo. Además se ilustran los diferentes tipos de interacciones que se tienen en los sistemas groupware.

Las interacciones que produce son:

1. **Interacción cara a cara:** Es un trabajo que se hace en el mismo lugar y en mismo tiempo, como una reunión de trabajo.

2. **Interacción asíncrona:** Es un trabajo que se hace en diferente tiempo pero en el mismo lugar. Puede ser un pizarrón electrónico donde se dejen notas por hacer. Esto es usado en la realidad virtual.

3. **Interacción distribuida sincronía:** Es un trabajo que se hace en el mismo tiempo pero en diferente lugar. Tal es el caso de las videoconferencias.

4. **Interacción distribuida asíncrona:** Es un trabajo que se hace en diferente tiempo y en diferente lugar. Es la edición colaborativa con el trabajo de varios autores.

La matriz anterior se parece a la realizada a Johanson [Johanson,80]:

Matriz de Johanson		
	En el mismo Lugar	En diferente Lugar
Comunicación Síncrona (al mismo tiempo)	Interacción cara a cara Salones electrónicos de juntas Sistemas para toma de decisiones en grupo Pantallas de computadora compartidas	Interacción remota Conferencias telefónicas Videoconferencia Pizarrones electrónicos Editores colaborativos
Comunicación Asíncrona (a diferente tiempo)	Interacción asíncrona Administración de proyecto Área electrónica de mensajes	Comunicación, Coordinación Correo electrónico Administradores de Flujo de Trabajo Control de versiones Conferencias asíncronas

Figura 4 Matriz de Johanson

La matriz de Johanson es más completa que la matriz de Coleman (ver figura 3) [Coleman,95]. Se asemejan al dividir a los sistemas en dos espacios, tiempo y lugar.

Como se ve el groupware es una parte muy importante en el campo de los Sistemas Distribuidos y más específicamente en el CSCW porque puede manejar el tiempo y el espacio.

1.5 EDICIÓN COLABORATIVA

La edición colaborativa es el poder trabajar con varias personas que se encuentran en diferentes partes del mundo para llegar aun fin común. La conciencia de grupo entra en este apartado debido a que se debe de estudiar como poder hacer dinámicas de grupo para que las personas que trabajan en un mismo objetivo puedan organizarse, coordinarse.

Lo que se busca al trabajar con la edición colaborativa es crear documentos fuertemente estructurados y en un mínimo de tiempo.

Se detallará más este tema en el capítulo cuatro del trabajo de investigación.

Dentro de la edición colaborativa, se encuentra un problema que afecta a los documentos que se refiere al control de un historial de versiones, del cual se hablara más adelante.

1.6 CONTROL DE VERSIONES

Después de ubicarnos dentro del campo de la edición colaborativa uno de los problemas que se presentan en este campo es la administración de versiones que tiene como objetivo hacer el seguimiento y permitir el acceso a un historial de estados importantes de un recurso distribuido (en este caso un documento). El control de versiones se presenta cuando se trabaja en una edición colaborativa, varios autores trabajan sincronamente o asincronamente sobre un documento a la vez.

El desarrollo paralelo proporciona disponibilidad adicional de un recurso en ambientes multiusuarios y distribuidos, además le permite a los autores hacer cambios en el mismo recurso al mismo tiempo, y juntar estos cambios en alguna fecha posterior. La administración de las configuraciones ataca los problemas de hacer el seguimiento y permitir el acceso de múltiples recursos como un conjunto de recursos, no simplemente como recursos individuales.

El control de versiones permite a diferentes versiones coexistir en el mismo documento, capacitándolos individual y compartidamente [Vitali&Durand,94].

La consistencia es el problema inicial del control de versiones que significa la dependencia de los estados de los vínculos de un documento con su referencia. Cuando un documento es editado o revisado, sus localizaciones cambian y son destruidos los registros que indican que el texto fue visto. Si el vínculo por su parte cambió por el correspondiente nuevo documento, significa que probablemente el vinculo original cambio y por consiguiente es probable que se pierda la información de la antepenúltima versión del documento.

Es importante saber quién es el responsable del documento, pero tiene mayor relevancia en el caso en que el documento está redactado por múltiples autores, debido a que se crean versiones sucesivas de un mismo documento, pero de diferentes autores, y si no se lleva un control de quién autorizó el cambio, cuándo lo autorizo, etc., puede ocasionar serios problemas al momento de juntar las partes del documento, debido a que no tendrá una

coherencia de la información. Por eso es importante llevar un control de las versiones de la historia del documento para saber cuál es la última y quién la hizo.

Una característica de las versiones es que las versiones son inmutables debido a que no se pueden hacer cambios a la vieja versión debido a que ésta es guardada en el repositorio (Base de Datos) donde se ve el historial del mismo.

El control de versiones provee una solución al problema de direcciones (*addressability*). Debido a que las versiones son inmutables, las versiones continúan con su dirección todo el tiempo. Esto también significa que ninguna estructura de esquema de dirección puede ser usada para referirse dentro de un documento versionado.

Se han planteado diferentes modelos para poder resolver el problema del control de versiones. Los modelos que se plantean son los siguientes:

El modelo Palimpsest [Capítulo 2] cuyo escenario básico se da cuando varias personas trabajan al mismo tiempo en un mismo documento, por consiguiente, se requiere que se hagan cambios sin tener que avisar a los otros autores en el mismo momento de los cambios, sin que estos cambios alteren la armonía y la consistencia del documento. El modelo describe estructuras de datos compartidos, también define una gran variedad de tipos de deltas (procesos). El modelo maneja algunas directrices básicas como:

- Cuando un proceso no tiene comunicación, los datos quedan sin modificaciones.
- Mayor flexibilidad en resoluciones de conflictos entre acciones.
- El modelo provee de un constructor general de tipos de datos.
- Por la descripción, el modelo provee de una gran familia de tipos de deltas.

El modelo maneja dos niveles que son: un nivel del tipo de dato y un constructor de datos. Además el modelo maneja para la consistencia del documento, cuando se ha modificado, tres niveles que se encargan de que no haya ambigüedad, detección de conflictos y da resoluciones.

En una simple estructura de versiones, algún número de documentos versionados pueden ser creados, y los nuevos deltas pueden ser agregados en cada versión, debido a que las versiones son representadas en Palimpsest como objetos, y una aplicación no se preocupa acerca de todas las deltas individuales. En vez, Palimpsest puede ser direccionado para ajustar el nivel de los tipos de datos incluyendo solamente los datos que correspondan a las deltas de una versión particular.

Otro modelo es el de Chimera [Capítulo 2] que propone un método de administración de versiones de estructuras de hipertexto los cuales se almacenan en forma separada de los objetos versionables. Cuando se separan las estructuras de hipertexto y los objetos en diferentes repositorios, podría parecer más complicado el manejo de éstos, pero no lo es, lo

que sucede es que muestra más claramente la separación de las responsabilidades entre las estructuras de versiones de hipertexto y los objetos versionados.

Su arquitectura permite el soporte de hipertextos versionados, soporta ambientes heterogéneos de almacenaje de objetos que contiene un sistema manejador de versiones, para este soporte se necesitan varios requerimientos como el soporte de objetos versionados y los no-versionados, versiones para los conceptos de los hipertextos (ligas y anclas), permitir las ramificaciones y la habilidad de retroceder en la obtención de versiones de un documento.

El modelo Chimera se compone de los siguientes conceptos:

- Los objetos son los indicadores de la persistencia de los objetos almacenados afuera del sistema.

- Las vistas nombran las entidades activas que muestran los objetos. Las vistas dentro del modelo son simplemente indicadores de una vista ejecutada y almacena su código dentro de un repositorio. La vista es una combinación de una vista y una presentación de un objeto. Las vistas contienen anclas.

- Una ancla es una porción de una vista como un articulo de interés, las anclas se definen y son administradas por las vista.

- Una liga es un conjunto de anclas.

Otro modelo que estudia cómo resolver el problema de versiones es el de Contexto Anidados [Capítulo 2]. El modelo define a los documentos estructurados como la incorporación de los conceptos como encapsulación, modulación y abstracción, en hipermedia los documentos utilizan la multimedia como base en los modelos conceptuales con nodos (componentes de los documentos) y ligas o vínculos (relaciones de los nodos). Lo anteriormente escrito es la parte central de este modelo.

Este modelo se extiende y distingue dos clases básicas de nodos, llamados nodos terminales y compuestos, el último es el concepto central del modelo. Cada nodo tiene un conjunto de anclas que actúan como la interfaz externa del nodo. Las anclas encapsulan la definición de regiones. Cada ancla tiene asociado un id y un valor. Las anclas son usadas como puntos finales de las ligas, y para especificar eventos. Un nodo terminal contiene dato cuya estructura interna, si existe alguna, es dependiente de la aplicación y no será parte del modelo. Un nodo compuesto agrupa entidades, llamadas componentes, incluyendo otros nodos compuestos. La clase de nodos compuestos puede ser especializados dentro de otras clases, incluyendo la clase de nodos de contexto.

En el modelo un nodo es una entidad que tiene como atributos un contenido y una lista de anclas. Cada elemento en la lista de anclas se le llama ancla con nodo y define una región en el contenido de éste. Además a los nodos se les puede definir que como los que contienen fragmentos de información. Un evento es definido por la presentación de un

conjunto marcado e información de un nodo o por la selección de eventos o por el cambio de atributo de un nodo. Un evento puede estar en uno de los siguientes estados: preparar, preparado, ocurrido, pausa y espera; sin embargo, cada evento tiene un atributo asociado en el nodo donde esté definido.

NCM extiende los trabajos previos permitiendo que copias distintas (las representaciones) del mismo pedazo de información (en los mismos o medios diferentes) se traten como versiones del mismo pedazo de información. Esto extendió el uso de la noción de versión, acoplado con un mecanismo de notificación, provee una base buena para el trabajo cooperativo.

En NCM, el único nodo de contexto de usuario y nodo son los temas del control de versiones. Cada atributo (incluyendo el contenido) de un contexto de usuario o el nodo contenedor puede especificarse como versionable o no-versionable. El valor de un atributo no-versionable puede ser modificado sin crear una nueva versión. Las modificaciones en los valores de los atributos de los si versionables tienen que ser sobre una nueva versión del objeto. Por supuesto, algún tipo del mecanismo de notificación se necesitará mejorar para apoyar la versión, especialmente en el caso de actualización concurrente de atributos de los no-versionable. Además, en NCM, el usuario puede especificar si la adición de nuevos atributos al nodo se permite sin crear una nueva versión. El modelo introduce la noción de estado de un nodo contenedor y un nodo de contexto del usuario para controlar consistencia a través de nodos correlacionados, para apoyar trabajo cooperativo y para permitir creación automática de versiones. El estado es simplemente un nuevo atributo del nodo cuyo valor afecta, y es afectado por, la ejecución de operaciones seguras.

Para dirigir el problema de mantener la historia de un documento, el método introduce la clase de nodos de contexto de versión. Un contexto de versión V es un nodo de contexto que contiene contextos únicos del usuario y los nodos contenedores, que son datos u objetos de almacenaje de estos. V agrupa nodos que representan versiones de datos de la misma entidad, con un nivel de abstracción, sin implicaciones necesarias que una versión se deriva de otra. Los nodos en V se llaman las versiones correlacionadas, y ellos no necesitan pertenecen a la misma clase de nodos. La derivación de la relación es capturada explícitamente por los nexos en V. Nosotros decimos que V_2 se derivó de V_1, si hay un nexo en V_2 a V_1 en V. Las anclas en este caso simplemente dejan de ser precisos respecto a cual es la parte de V_1 que genera la parte de V_2. Un contexto de versión induce un diagrama estructurado no conectado en todas las versiones. No hay restricción sobre nexos, excepto en la derivación de las relaciones que deben ser acíclicas. Estas versiones que son de relaciones acíclicas no son incluidas en los contextos de versiones que maneja el modelo.

Un usuario puede manualmente agregar nodos (explícitamente indicando que éstas son las versiones del mismo objeto) y los nexos (explícitamente indicando como las versiones que se derivan) al contexto de versión, o él puede crear un nuevo nodo desde otro para invocar una operación de versiones, que entonces automáticamente actualizará el contexto apropiado de la versión. La creación de los nexos de derivación automáticamente ocasiona al objeto de predecesor para ser comprometido a fin de la consistencia de conserva.
El modelo maneja la propagación de versiones. Cuando la propagación de versiones es automática se puede causar un gran problema debido a la creación de un número

indeterminado de versiones de un mismo documento. Lo que hizo el modelo NCM fue poner un límite de propagaciones de un documento y preguntar si se hace una propagación automática o no.

Trabaja sobre un documento implícito en la creación de nuevas versiones de todos los usuarios.

Otro modelo es el de WebDAV [Capítulo 2] el manejo de versiones en este modelo especifica un conjunto de mecanismos que pueden ser explotados para establecer un conjunto de políticas que permitan a las aplicaciones de los clientes y usuarios encontrar un balance apropiado entre sus necesidades.

WebDAV provee muchos beneficios como:

- Permite que todas las publicaciones puedan ser contenidas en el Web. Ahora los grupos y los individuos pueden usar el protocolo Http para publicar directamente sus trabajos.
- Los trabajos en grupos pueden colaborativamente crear documentos en algún sobre usando cerraduras para prevenir conflictos de sobre escritura. Debido a la naturaleza distribuida en como se comporta la red, los grupos de trabajo pueden tener miembros que no sean de la misma organización.

El WebDAV da unas reglas semánticas para el manejo del control de versiones que son:

- Un recurso versionado es una abstracción de un recurso, el cuál esta sujeto a control de versiones.
- Una revisión es una versión particular de un recurso versionado. Una revisión inmutable es una revisión que una vez creada, nunca puede ser cambiada sin crear una nueva revisión.
- Una revisión mutable es una revisión que puede cambiar sin crear una nueva versión.
- Una revisión inicial es la primera revisión de un recurso versionado y no tiene predecesores dentro del recurso versionado.
- Las revisiones tienen: un nombre de revisión, un identificador de revisión y una etiqueta.
- Un predecesor de una revisión es una revisión de la cual ésta revisión es creada.
- Un sucesor de una revisión es una revisión derivada de esta revisión. Una revisión puede tener un predecesor y múltiples sucesores. La relación es-derivado-de entre revisiones de un recurso versionado forman un árbol.

- Una historia de revisiones es una representación concreta de los elementos de un recurso versionado incluyendo todas las relaciones de predecesores y sucesores, nombres de revisión, etc.

- Una línea de descendentes es una secuencia de revisiones conectadas por relaciones sucesor / predecesor desde la revisión inicial a una revisión específica.

- Una actividad es un recurso que se refiere a un conjunto de revisiones nombradas que corresponden a alguna unidad de trabajo o cambio conceptual.

- Un espacio de trabajo es un recurso que es usado para determinar que la revisión de un recurso versionado debe ser accesado cuando el recurso es referenciado sin un nombre de revisión particular.

- Una regla de selección de revisión especifica que revisión de un recurso versionado debe seleccionarse.

- Una configuración es un conjunto de recursos relacionados nombrados donde cada miembro se refiere a una revisión específica de un recurso versionado.

Estos modelos son creados para que se puedan manejar las versiones de los documentos, en historiales utilizando conceptos del WEB como ligas, anclas, etc. Los modelos se explicaran específicamente más adelante.

Para el manejo del control de versiones, se han desarrollado sistemas que se dedican a la administración de las versiones, se dará una breve explicación de los sistemas y son:

El Sistema MUCH que fue creado en la Universidad de Liverpool para permitir a un grupo de personas trabajar colaborativamente en un ambiente de hipertexto e hipermedia. Su arquitectura consiste en una red semántica, una ingeniería transversal y una interfase jerárquica de la red semántica generada por el programa transversal. El sistema soporta la escritura colaborativa asíncrona, tiene una arquitectura cliente – servidor. Dentro del sistema los nodos son referencia a pedazos de información que al juntarlos se crea un documento estructurado.

Otro sistema que se tomo en cuenta para esta investigación fue el de CLEARCASE, realizado por Silicón Graphics, fue creado para apoyar el control de versiones, presenta la capacidad de manejar múltiples versiones de software desarrollado. Permite relacionar documentos y la información de las bases de datos, usando pistas para las versiones en un software en construcción, ejecutando programas individuales o en todas las liberaciones de los acuerdos de la definición de los usuarios para las versiones especificadas y forzando en los lugares desarrollados específicamente.

El sistema utiliza herramientas que son comunes en nuestra vida diaria como son: WordPerfect 5.0 para UNIX, Configuraciones de Script para los CGI y formatos de Mosaicos para bases de datos.

Además este sistema resolvió problemas relacionados con el control de versiones:

- Indexación
- Formas / Escrituras
- Anotaciones
- Estructuras de archivos y conversiones de nombres.
- El factor humano
- Compatibilidad
- Vinculaciones

Estos puntos serán vistos con más detalle en el capítulo tres.

Otro sistema a tomar en cuenta será el de CVS, se podría decir que éste es el más común debido a que se puede bajar gratis de Internet, por consiguiente se ha conseguido mucha información con respecto a este sistema, el sistema muestra un historial de versiones de un documento. Mantiene un control de las versiones dentro de un historial, es decir, una base de datos. El sistema además de mantener un control de las versiones, gestiona las peticiones de Entrada y Salida de la información dentro de la base de datos.

La base de datos o Histórico se dedica a registrar los accesos y las salidas de la información llevando un registro de los mismos. El CVS esta basado en RCS que es un sistema de control de revisiones. Con referencia a su seguridad maneja controles que no permite que un documento o mejor dicho una versiones se modifique dos veces sin crear una nueva versión, y otros puntos importantes que se verán en su capítulo.

El último sistema versará en torno al de HYPERPROP que es el único sistema que está basado en un modelo que se estudia en el siguiente capítulo que es el NCM (modelo de Contexto Anidados), el sistema maneja tres vistas que controlan diferentes aspectos y son:

1.- Vista Estructurada: Soporta búsquedas, da características de los nodos y las ligas.

2.- Vista Temporal: Responsable del soporte de las especificaciones de las relaciones temporales de un hiperdocumento.

3.- Vista Espacial: Soporta la definición de las relaciones de espacio entre los componentes de un documento.

En su capítulo correspondiente se tratara más detalladamente los conceptos de cada uno de los sistemas que anteriormente se han explicado brevemente.

CAPÍTULO 2

MODELOS DE VERSIONES

En este capítulo se hablará sobre los modelos que se han desarrollado para resolver el problema de la administración de versiones de documentos.

2.1 MODELO PALIMPSEST

El modelo Palimpsest fue diseñado para poder manejar los diferentes problemas de versiones que podrían ocurrir en la manipulación colaborativa de datos compartidos. Puede representar muchas aplicaciones con estructuras de datos, incluyendo vínculos, por el momento, el modelo puede describir archivos de datos lineales, estructuras de hipertexto, tablas, registros de formato mixto (donde los registros pueden ser números y letras), y también permite la descripción de un rango amplio de estrategias de control de versiones incluyendo[Durand,94]:

- Su escenario básico se da donde varias personas trabajan en un mismo documento, lo que se requiere es que pueden hacer cambios sin tener que comunicarse, permitiendo que los cambios no afecten la integridad y la armonía de los mismos. Las plataformas donde puede trabajar son:

  - Redes de área local de alta velocidad (Fast LANs): Utilizando estaciones de trabajo (Workstations).
  - Redes de área metropolitana de velocidad media (Medium – speed WAN): Utilizando el acceso directo que se tiene a Internet.
  - Conexiones por medio Módems a Internet.

El modelo describe la estructura de datos compartidos, accesados por un número arbitrario de procesos marcados para cambiar e incurrir arbitrariamente las esperas antes de los cambios hechos por un proceso. Cada proceso tiene una vista diferente de la estructura de datos compartidos.

El modelo Palimpsest permite la definición de una gran variedad de tipos de deltas (procesos), cada una corresponde a una aplicación de transacción. Los estados de los documentos están definidos por un conjunto de deltas. Las deltas no son ordenes secuenciales como las instrucciones de edición, pero puede ser interpretadas sin referencia a una secuencia de tiempo explícito. Las transacciones que hacen los deltas son:

- Creación de deltas, indicado con la letra C al principio de la palabra, que guarda la creación de una cadena de caracteres.
- Inserción de deltas. Indicado con la letra I al principio de la palabra, que guarda la inserción de un texto dentro de una cadena de caracteres.

- Borrado de deltas, indicado con la letra D al principio de la palabra, que borra el registro del texto desde la cadena de caracteres.

El modelo contiene cuatro principios básicos que son [Durand,94]:

- Cuando un proceso no tiene comunicación, los datos en este modelo permanecen inmutables. Cuando la sincronización entre las actualizaciones no puede ser garantizada, el único camino es remover lo escrito, tomando en cuenta que la actualización es la última aportación del escritor al documento.

- Mayor flexibilidad en resoluciones de conflictos entre acciones, el modelo expresa la estructura de datos enteramente en términos de deltas o mejor dicho en acciones de un proceso determinado.

- Aplicaciones que requieren una gran variedad de tipos de datos, Palimpsest provee un constructor general de tipos de datos y tipos suficientes para poder representar el gran rango de estructuras de datos que se pueden contener en un documento.

- Aplicaciones que requieren una gran variedad de tipos de transacción, el modelo Palimpsest esta provisto por la descripción de una gran familia de tipos de deltas, cada una consiste de un paquete de cambios básicos.

El modelo está constituido por dos niveles. El nivel del tipo de dato es un conjunto de tipos primitivos y un constructor de tipos el cual representa las aplicaciones de los tipos de datos. Con el nivel de tipos de datos el archivo es una secuencia de caracteres. El nivel describe la estructura de los estados de las aplicaciones de los datos independientes de las operaciones que tiene el dato.

El nivel de representación de Palimpsest describe constructores de los niveles de los tipos de datos como un conjunto de deltas, cada conjunto graba algunos aspectos de la historia de los datos, esto es donde el dato es creado y actualizado, mientras el nivel del tipo de dato es una vista del estado de la aplicación del dato obtenida por la interpretación del modelo. Mientras la aplicación naturalmente tendrá que repartir con el nivel de representación, al grado que permita la interpretación y manipulación del historial de los cambio.

El modelo define niveles de coherencia los cuales aseguran que un conjunto de datos pueda ser interpretado sin ambigüedad alguna.

El modelo define tres niveles de consistencia permitiendo cambios con detección de conflictos y dar resoluciones. La noción fundamental de la consistencia es la consistencia mínima. Un conjunto de deltas tiene consistencia mínima si estos miembros se refieren únicamente a las deltas dentro del conjunto. Esta es una condición fácil de satisfacer: la creación de deltas inicia desde un conjunto con consistencia mínima y refiere a las deltas sin que el conjunto pueda ser adicionado en la producción de un conjunto con consistencia mínima, es decir, al dar la consistencia a un conjunto de deltas, un proceso puede hacer nuevos deltas y preservar la consistencia mínima sin preocuparse acerca de otros procesos.

La consistencia mínima la requiere el modelo para la semántica de la información que no tienen deltas y no están definidas.

Provee de constructores de tipo, para tipos de conjuntos, tablas, ligas o tipos primitivos. Los tipos primitivos pueden cubrir una gama amplia de objetos como: para datos de textos, ellos podrían ser códigos de caracteres escogidos desde un conjunto de caracteres específicos; para datos gráficos, el valor de color del píxel; y para las hojas de cálculos los valores de las celdas si son cadenas caracteres (string) o numéricos.

Las deltas también contienen bloques de datos, los cuales representan una porción de algunos datos en el sistema. Un bloque de dato contiene datos constantes de algún tipo primitivo o creado por una composición de un constructor de tipo y un tipo básico.

Palimpsest provee de estructuras primitivas representando la organización de los tipos de las bases de datos y estos pueden ser [Durand,94]:

- Deltas: Son referencias que permiten al modelo la descripción de la versión y el cambio de datos para un propósito de una aplicación.

- Sets: Es un constructor de tipos y representaciones de un grupo de objetos desordenados.

- Secuencias: Es la representación ordenada de un conjunto de artículos. Sus operaciones son: inserción, eliminación y mover un elemento.

Maneja Tablas n-dimensionales definidas en estructuras tabulares por n-secuencias o sets. Los cuales especifican las dimensiones de la tabla. Las operaciones que pueden ejecutarse son: la adición o modificación de las dimensiones de la tabla, asociación de objetos con una posición en la tabla o el reemplazo del contenido de una posición en la tabla.

Los grados de las tablas se definen en coordenadas en el espacio. Una celda individual es una combinación de elementos, unidos por cada axisa de la tabla. En dos dimensiones, por ejemplo, los elementos de las dos axisas son representativos de sus columnas o filas enteras. El contenido de las celdas son copiadas separadamente de la tabla. El Palimpsest permite la asociación del mismo objeto con varias posiciones de la tabla, muchos como una secuencia de caracteres podrían contener la misma letra en diferente posición dentro de la tabla.

La implementación del modelo se realiza en un grupo de servidores de comunicación de punto. Cada servidor tendría una base de datos local de deltas, y estos podrían comunicarse con otros servidores Palimpsest, cambiando deltas que reflejan las actividades de los usuarios. El modelo no impide el uso de la arquitectura cliente – servidor, el modelo no requiere de un sincronizador central para poder manejar las transacciones.

La arquitectura es relativamente simple, ésta mantiene tres estructuras de datos que son las siguientes:

- Un registro cronológico es único para cada delta conocido a este nodo.

- Una base de datos de deltas optimizada para el acceso rápido de aplicaciones.

- Una lista de nodos con los cuales cambiaría actualizaciones.

Un ejemplo de una descripción de Palimpsest [Durand,94] para una aplicación que provee una línea simple de nodos textuales, con ligas o vínculos entre segmentos arbitrarios de nodos sería:

```
Deltatype text-node is {
    Data contents: Sequence of
Characters;
    Provide (contents);
}
Deltatype copy is {
        Range x: Sequence of Characters;
        Address y: Sequence of Characters;
        Copy ( x, y );
}
Deltatype move is {
        Range x: Sequence of Characters;
        Address y: Sequence of Characters;
        Copy ( x, y );
        Remove (x );
}
Deltype insert is {
   Data insertion-contents :  Sequence of Characters;
   Address where : Sequence of Characters;
   Copy ( insertion-contents, where );
}
Deltatype deletion is {
    Range x: Sequence of Characters;
    Delete ( x );
}
Deltatype link is {
  Data link_contents = Sequence of {
            Data from : Reference;
            Data to : Reference;
            Data explainer : Sequence of Characters;
    }
}
```

En cada delta se declara un número de tipo de dato y el nombre del dato de dirección, seguido por una lista de acciones básicas ejecutadas por los deltas. Las acciones son: inserción de texto, copiado de texto, borrado de texto, movimiento de un texto de una localidad a otra y creación de ligas.

Un simple control de versiones puede facilitar agregar algunos nuevos deltas. Un delta de versión se encargaría de registrar la creación de una nueva versión, que es simplemente un conjunto de deltas, cada uno representando un cambio que componen la versión. Un esquema de una adición de un delta de versiones es el siguiente:

```
Deltatype versión is {
        Data versión-contents is Séquense of   {
                Data version-changes : Set of Delta-references;
                Data version-name : Sequence of Characters;
         }
         provide ( version-changes);
}

Deltatype add-change is {
       Address the-version : version.version-contents
       Data new-deltas : Set of Delta-references;
        Copy ( new-deltas, the-version);
}
```

En una simple estructura de versiones, algún número de documentos versionados pueden ser creados, y los nuevos deltas pueden ser agregados en cada versión, debido a que las versiones son representadas en Palimpsest como objetos, y una aplicación no se preocupa acerca de todas las deltas individuales. En su lugar, Palimpsest puede ser direccionado para ajustar el nivel de los tipos de datos incluyendo solamente los datos que correspondan a las deltas de una versión particular.

2.2 MODELO CHIMERA

Propone un método de administración de versiones de estructuras hipertexto, que se almacenan en forma separada de los objetos (documentos) versionables. Aquí se incluye la necesidad de mantener subconjuntos de estructuras de hipertexto las cuales son afectadas por ediciones a las versiones de objetos dadas, y un mecanismo para asociar este subconjunto de estructuras hipertexto a las versiones de objetos almacenadas externamente. Se incluye el concepto de tabla de asociación entre versión y configuración [Whitehead et al,94].

Las estructuras de versiones de hipertexto en situaciones donde las estructuras son mantenidas en un repositorio y los objetos relacionados inicialmente se ubican en uno o varios repositorios separados, esto es significativamente más complicado que el caso donde ambas las estructuras de hipertexto y los objetos relacionados están en el mismo repositorio. Sin embargo, al separar las estructuras de hipertexto de los objetos permite la clara separación de responsabilidades entre las estructuras de versiones de hipertexto y los objetos versionados.

La arquitectura de Chimera permite el soporte de hipertexto versionable en ambientes heterogéneos de almacenaje de objeto que contiene un sistema de manejo de configuración.

El soporte de las versiones de hipertexto para un ambiente compartido heterogéneo de almacenaje de objetos, sus requerimientos son más que los identificados para los casos no-heterogéneos. Los requerimientos son: soporte para objetos versionados y los no-versionados, versiones para todos los conceptos de los hipertextos (como ligas y anclas), permitiendo ramificaciones de múltiples versiones y la habilidad de retroceder en la obtención de versiones de un objeto (documento).

Los requerimientos adicionales para los casos de almacenaje de objetos heterogéneos son:

- Los objetos almacenados externos al sistema de hipertexto y las estructuras de hipertexto almacenadas internamente en el sistema deben ser capaces de desarrollarse independientemente. Los conceptos de hipertexto son capaces de manipular independientemente al sistema externo o viceversa. Cuando las versiones de hipertexto están fuera de sincronía con respecto a los documentos de versiones, el sistema debe hacer que las versiones del hipertexto se acoplen al tiempo de los documentos.

- Esto es posible si se tienen múltiples nombres y subconjuntos de versiones del total de las estructuras de hipertexto. Usando los subconjuntos, esto es posible la captura de porciones de estructuras de hipertexto que podría ser afectado por un cambio del objeto (documento).

- Da un método para asociar una versión externa del documento al subconjunto de estructuras de hipertexto y viceversa. Esta asociación provee un puente entre un documento externo versionado y la estructura de la versión del hipertexto, permitiendo una versión determinada del documento externo antes de ser combinada con la estructura de la versión.

- Las versiones de estructuras de hipertexto podrían ocurrir como posibles transparencias para el usuario. El sistema de versiones podría no impulsar constantemente al usuario para los nombres de las versiones, a cambio de la selección automática y asignación de nombres de versiones.

- Las demandas de herramientas externas podrían ser minimizadas. Los sistemas de servidores de hipertexto asumen que algunos niveles de modificaciones de las herramientas serían requeridas para el uso de las funcionalidades de los servidores de hipertexto. Sin embargo, estas modificaciones podrían ser tan mínimas como sea posible para la facilidad de la herramienta integrada.

- Un repositorio externo de objeto debe ser capaz de recobrar una versión arbitraria de un documento. Esto es necesario, porque una liga puede contener un ancla que refiere a una versión más vieja del objeto actual que debe recobrarse para completar una liga transversal.

- Un repositorio externo de documentos debe proveer una función que, dado un objeto, devuelve su versión. Esta función necesita identificar una versión del objeto tan que la correcta versión pueda usarse mientras se está trabajando con el documento.

- Un depósito externo de documentos podría proveer una función la cual regresaría inmediatamente el predecesor de la versión de un objeto. Esta función es empleada por el sistema de hipertexto para determinar si un objeto externo ha sido verificado y su estructura de hipertexto ha sido accesada.

El modelo Chimera se emplea en un servidor de hipertexto el cual provee un almacenaje persistente de una estructura, funciones para manipular las estructuras y soportar las ligas dentro de las estructuras. La diferencia clave del modelo Chimera y otros modelos de versiones, es que Chimera no ocupa el almacenaje de las estructuras de hipertexto dentro de objetos relacionados por la estructura, y Chimera asocia anclas con vistas de objetos, no los objetos en si mismos. Otra diferencia es que soporta múltiples vistas simultáneamente de un objeto, tal como lo hace las hojas de cálculo en los que tenemos varias vistas en celdas y en gráficos de un mismo objeto.

El modelo Chimera se compone de los siguientes conceptos:

Los objetos son los indicadores de la consistencia de los objetos almacenados afuera del sistema. El término de objeto típicamente no se refiere al objeto mismo, más bien a su indicador del objeto.

Las vistas nombran las entidades activas que muestran los objetos. Las operaciones típicas son provistas para las vistas, incluye examinar, crear, y redactar objetos dentro de su dominio. Las vistas dentro del modelo Chimera son simplemente indicadores de una vista ejecutable, y la Chimera no almacena el código ejecutable en su depósito.

Una vista es la combinación de una vista y la presentación de un objeto. Un objeto puede ser mostrado por más de una vista y estos participan en múltiples vistas. Las vistas contienen anclas.

Una etiqueta de ancla da una porción de una vista como un artículo de interés. Las anclas se definen y son administradas por las vistas dentro de un marco de una vista. Las anclas son típicamente no almacenadas en los objetos en sí mismos.

Una liga es un conjunto de anclas y de aquí en adelante contiene anclas. Las ligas relacionan porciones de vistas.

Un cambio que se requiere para poder comprender mejor al sistema Chimera es la adicción de la dimensión del tiempo para poder marcar todos los conceptos sobre las versiones. El modelo utiliza las graficas acíclicas directas (GAD) como componentes de las versiones. El GAD ocurre en los sistemas de versiones que permiten el desarrollo paralelo. Las estructuras del árbol ocurren cuando el desarrollo permite la ramificación con una bifurcación en el desarrollo de la ruta ocurrida en uno o varios objetos (documentos).

Una ramificación independiente es una ventaja de un árbol de GAD que puede ser completamente separada de la ramificación original, es fácil de identificar en una ruta de un desarrollo paralelo. En el desarrollo del código, las ramificaciones independientes son usadas para los códigos de reúso en situaciones donde un objeto podría ser derivado de una versión existente. Las versiones son llamadas usando el reverso para numerar el esquema. Con este esquema, se muestra como es la función para la siguiente versión. Se asume una versión inicial con un número de 1 si una ramificación es hecha desde un objeto, se adiciona el ".1" a la cadena al final del objeto. Si un nuevo hermano es derivado desde un objeto, se adiciona 1 como último digito de la versión. Obteniendo la versión "1.1", la función para la siguiente versión regresa "1.2", una ramificación de la versión "1.2" tomaría la asignación de "1.2.1"

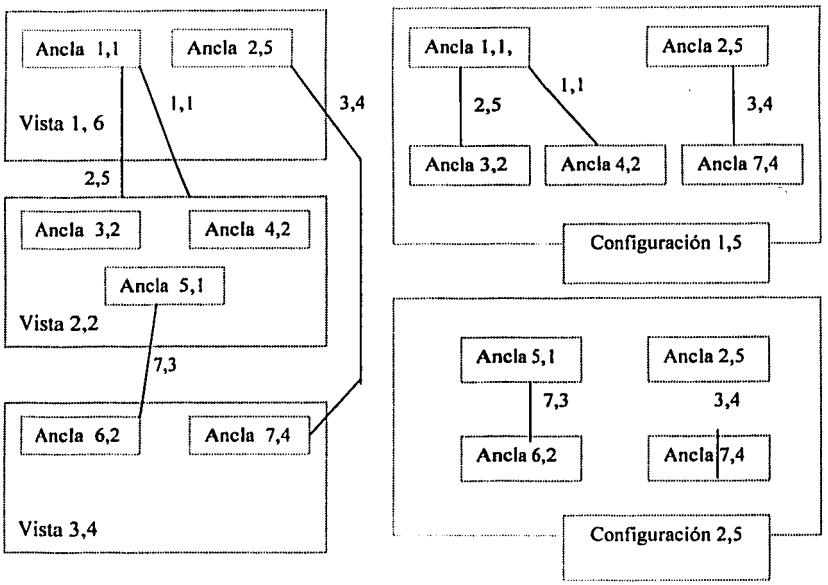


Figura 5 Configuración de una ramificación en el modelo Chimera

La figura 5 se trata de una configuración, del lado izquierdo se muestra una estructura de hipertexto que muestra tres vistas, las anclas definidas sobre las vistas y las líneas que simbolizan ligas de los contenidos de las anclas. Los pares de números son del concepto de identificación, y son para las versiones, la pareja de números próximos a las líneas representan las ligas. Del lado derecho, la Configuración 1,5 contiene todas las modificaciones de los objetos de hipertexto que son afectados por el objeto de la vista 1,6. La configuración 2,5, muestra el mismo subconjunto de la estructura de hipertexto para la Vista 3,4.

El propósito del concepto de configuración es para el nombre, la colección, y los subconjuntos de versiones de una estructura de hipertexto que podrían ser afectadas por las modificaciones por un objeto externo. Con estos nombres, los subconjuntos de versiones, son fáciles de expresar las versiones de objetos externos. Una configuración marca lo conveniente para la creación de nuevas ramificaciones de un subconjunto de hipertexto sobre la creación de un nuevo objeto versionado. Otro requerimiento es sobre la posibilidad de capturar el documento original teniendo los múltiples nombres y los subconjuntos de las versiones.

La configuración soporta varias operaciones sobre sí mismo y sobre sus miembros. Es posible crear y eliminar una configuración, como también adicionar o borrar los conceptos de una versión desde la configuración. Además se puede crear nuevas versiones dentro de la configuración y reemplazar el concepto usado por uno nuevo.

Para la transparencia el servidor Chimera mantiene un conjunto de configuraciones activas para cada usuario.

2.3 MODELO DE CONTEXTO ANIDADO

En la definición de los documentos estructurados se incorporan conceptos como encapsulación, modularidad y abstracción. En hipermedia los documentos utilizan la multimedia como base en los modelos conceptuales con nodos (componentes de documento) y ligas o vínculos (relaciones de los nodos). La introducción del concepto de composición trae la noción de documento estructurado, siendo el concepto central de NCM[6][Soares&Souza,98].

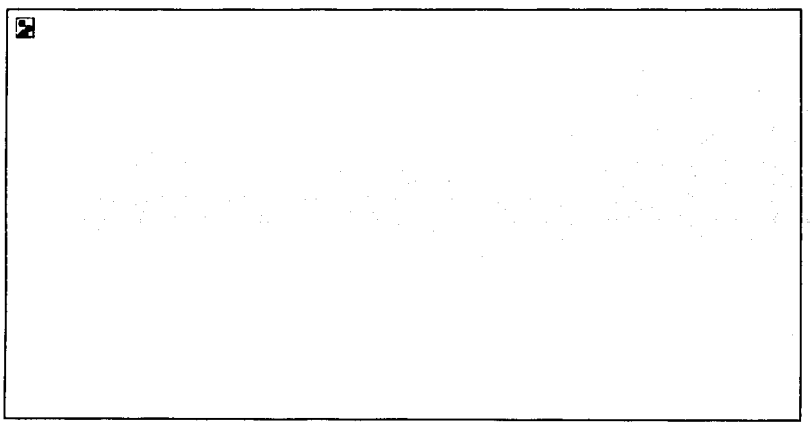


Figura 6 Clase de jerarquías en el Modelo de Contexto Anidado

[6] NCM acrónimo de Nexted Context Model –en español Modelo Contexto Anidado

La Figura 6, muestra que una entidad es un objeto que tiene como atributos un único identificador, un acceso controla la lista y una entidad descriptora. El único identificador tiene el significado usual. Para cada atributo de entidad, el acceso es uno. Nosotros frecuentemente usamos el nombre de atributo de una entidad para referirnos el valor de atributo. Cuando el contexto no permite esta simplificación, nosotros usaremos explícitamente el término del valor atribuido. La lista de control tiene una entrada que asocia un usuario o un grupo de usuarios a sus derechos de acceso para el atributo [Soares&Souza,94].

Un nodo es una entidad que tiene como atributos adicionales un contenido y una lista de anclas. Cada elemento en la lista de anclas se llama ancla del nodo y define una región en el contenido del nodo. Hay un ancla por default que representa el contenido entero de un nodo.

Este modelo se extiende y distingue dos clases básicas de nodos, llamados nodos terminales y compuestos. Un nodo terminal contiene dato cuya estructura interna, si existe alguna, es dependiente de la aplicación y no será parte del modelo. Un nodo compuesto agrupa entidades, llamadas componentes, incluyendo otros nodos compuestos. La clase de los nodos compuestos puede ser especializado dentro de otras clases, incluyendo la clase de nodos de contexto. Cada nodo tiene un conjunto de anclas que actúan como la interfaz externa del nodo. Las anclas encapsulan la definición de regiones. Cada ancla tiene asociado un id y un valor. Las anclas son usadas como puntos finales de las ligas[Soares&Souza,98].

Un ambiente autoritario debería ofrecer una buena edición y búsqueda de herramientas para definir los contenidos de los nodos y el contenido de la granularidad.

Las anclas son usadas para especificar eventos. Un evento es definido por la presentación de un conjunto marcado e información de un nodo o por la selección de eventos o por el cambio de un atributo de un nodo. Un evento puede estar en uno de los siguientes estados: preparar, preparado, ocurrido, pausa y espera. Sin embargo cada evento tiene un atributo asociado en el nodo donde esté definido.

Intuitivamente, tomando un suceso de presentación como un ejemplo (ve Figura 6), comienza en el estado de espera. Va al estado de prepara mientras algún pre-procedimiento de sus unidades de información está siendo ejecutadas. La palabra de usuario dentro de un marco de este papel tiene significados múltiples: significa un usuario en el sentido de una persona, una aplicación procesa o un programador de aplicación. Que es, cualquier cosa o nada, que hace al uso de los servicios definido en varias capas de interfase. Al final del procedimiento, el suceso va al estado preparado. A principios de la exposición de unidades de información va al estado ocurrido. Si la exposición se suspende temporalmente, el suceso permanece en el estado pausado, mientras la situación dura. Al final de la exposición, el suceso vuelve al estado preparado, cuando el atributo ocurrido es incrementado. Obviamente, sucesos instantáneos, como selección y atribución, la estadía en el ocurriendo estatal único durante un tiempo infinitesimal.

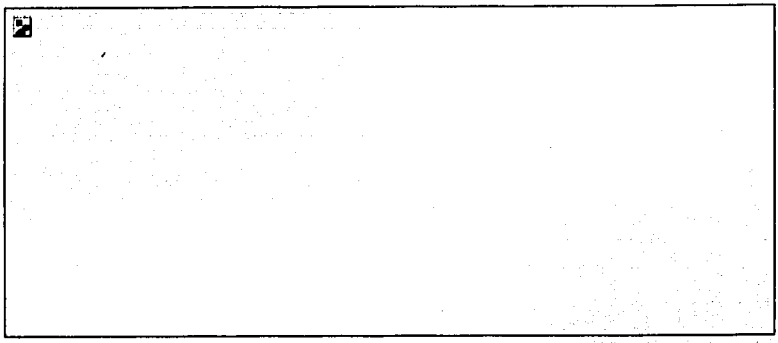


Figura 7 Mecanismo de los estados de los eventos

La figura 7 nos muestra como se realizan los eventos en los que puede estar un documento, de los pasos de reparar, actualización, pausa.

Como el modelo permite a los nodos ser recursivos contenido en diferentes composiciones y nodos compuestos ser anidados con alguna profundidad, esto es necesario para introducir el concepto de perspectiva. Intuitivamente, la perspectiva de la identificación de los nodos a través de secuencias de nodos compuestos anidados, el nodo inmediatamente es determinado en una primera observación. Formalmente, la perspectiva de un nodo N es una secuencia $P = (N_m,\ldots,N_1)$ con m^3 1, tal que $N_1 = N$, N_m es un nodo no contenido en ningún nodo compuesto en N_{i+1}, para I [1, m). N_1 es llamado el nodo base de P. Pueden existir varias perspectivas diferentes para un mismo nodo, si es que el nodo esta contenido en más de un nodo compuesto [Soares&Souza,98].

La ligas en el modelo de contexto anidado es una relación compuesta n : m por un fuente final del conjunto de puntos, un blanco final del conjunto de puntos y una reunión de puntos. Un nodo puede pertenecer a más de un nodo compuesto, la fuente y el blanco termina los puntos de un nexo son identificados por pares de la forma $<(N_k,\ldots,N_1), A>0$, tal es N_1 es un nodo, llamado nodo de ancla del nexo, N_{i+1} es un nódulo compuesto y N i se contiene en N_{i+1}, para todo i Î[1,k), con k>0, y A es un suceso de N_1. N_k es un componente del nodo compuesto, que contiene el nexo, o el nodo compuesto a sí mismo.

El punto de reunión define condiciones y acciones. Las condiciones podrían ser cumplidas por el punto de la fuente final en orden que las acciones sean aplicadas en los puntos del blanco final.

2.3.1 Control de versiones en el modelo de contexto anidado

Aunque la necesidad para el control de versión en sistemas de hipertexto es muy conocido, la complejidad de la interacción entra las versiones controladas y los otros requerimientos se ha demorado. NCM extiende los trabajos previos permitiendo que copias distintas (las

representaciones) del mismo pedazo de información (en los mismos o medios diferentes) se traten como versiones del mismo pedazo de información. Esto extendió el uso de la noción de versión, acoplado con un mecanismo de notificación, provee una base buena para el trabajo cooperativo [Soares&Souza,98].

En NCM, el único nodo de contexto de usuario y este nodo son los temas de control de versiones, como se observa en antecedentes gris en la Figura 6. Cada atributo (incluyendo el contenido) de un contexto de usuario o el nodo contenedor puede especificarse como versionable o no-versionable. El valor de un atributo no-versionable puede ser modificado sin crear una nueva versión. Las modificaciones en los valores de los atributos de los si versionables tienen que ser sobre una nueva versión del objeto. Por supuesto, algún tipo del mecanismo de notificación se necesitará mejorar para apoyar la versión, especialmente en el caso de actualización concurrente de atributos de los no-versionable. Además, en NCM, el usuario puede especificar si la adición de nuevos atributos al nodo se permite sin crear una nueva versión. Nosotros introducimos la noción de estado de un nodo contenedor y un nodo de contexto del usuario para controlar consistencia a través de nodos correlacionados, para apoyar trabajo cooperativo y para permitir creación automática de versiones. El estado es simplemente un nuevo atributo del nodo cuyo valor afecta, y es afectado por la ejecución de operaciones seguras.

Un nodo contenedor o un nodo de contexto del usuario N puede estar en uno los estados siguientes: comprometido, no comprometido u obsoleto. N está en el estado no comprometido sobre la creación. Cuando llega a ser estable, N puede promocionarse al estado comprometido o explícitamente por una operación específica o implícitamente por operaciones seguras dentro del modelo. Un nodo comprometido no puede directamente actualizarse o borrado, pero el usuario puede invocar otra operación específica para hacer lo obsoleto, permitiendo nodos que se derivan cuando son notificados.

Un objeto de guardado debe estar en el estado comprometido u obsoleto, donde el objeto esta en la capa de almacenaje son siempre en su forma final, que es no puede haber modificaciones en los atributos de los versionados.

Para dirigir el problema de mantener la historia de un documento, el método introduce la clase de nodos de contexto de versión. Un contexto de versión V es un nodo de contexto que contiene contexto único del usuario y los nodos contenedores, que son datos u objetos de almacenaje de estos. V agrupa nodos que representan versiones de datos de la misma entidad, con un nivel de abstracción, sin implicaciones necesarias que una versión se deriva de otra. Los nodos en V se llaman las versiones correlacionadas, y ellos no necesitan pertenecen a la misma clase de nodos. La derivación de la relación es capturada explícitamente por los nexos en V. Nosotros decimos que V_2 se derivó de V_1, si hay un nexo en V_2 a V_1 en V. Las anclas en este caso simplemente dejan de ser precisos respecto a la cual es la parte de V_1 que genera la parte de V_2. Un contexto de versión induce un diagrama estructurado no conectado en todas las versiones. No hay restricción sobre nexos, excepto en la derivación de las relaciones que deben ser acíclicas. Estas versiones que son de relaciones aciclicas no son incluidas en los contextos de versiones que maneja el modelo.

Un usuario puede manualmente agregar nodos (explícitamente indicando que estas son las versiones del mismo objeto) y los nexos (explícitamente indicando como las versiones que se derivan) al contexto de versión, o él puede crear un nuevo nodo desde otro para invocar una operación de versiones, que entonces automáticamente actualizará el contexto apropiado de la versión. La creación de los nexos de derivación automáticamente ocasiona al objeto de predecesor para ser comprometido a fin de la consistencia de conserva.

El modelo maneja la propagación de versiones. Cuando la propagación de versiones es automática se puede causar un gran problema debido a la creación de un número indeterminado de versiones de un mismo documento. Lo que realiza este modelo hace ponerse un límite de propagaciones de un documento y pregunta si se realiza una propagación automática o no.

Trabaja sobre un documento implícito en la creación de nuevas versiones de todos los usuarios.

2.3.2 HIPERBASES PÚBLICAS Y BASES PRIVADAS

La noción del nodo de contexto puede usarse para apoyar trabajos cooperativos como se indicará a continuación. Considera el conjunto de todos los nodos de contexto del usuario y contenedores de nodos para ser particionados en varios subconjuntos. Uno y únicamente uno de ellos formarán la hiperbase pública, que corresponde a la información estable pública. Los otros subconjuntos formarán las bases privadas, usadas para modelar interacción del usuario con un documento de hipermedia, según el paradigma propuesta por el Modelo de Dexter. Una base privada puede contener a las otras bases privadas, permitiendo la organización de una jornada de trabajo en varios sub-sesiones anidadas. Para especificar el nodo de contexto del usuario o contenido que pueda pertenecer a uno y uno únicamente de estas bases (públicas o privadas).

Se define a la hiperbase pública, denotada como HB [Soares&Rodríguez, 94], como un tipo especial de nodo de contexto que agrupa conjuntos de nodos contenedores del usuario y nodos, la HB agrupa todas las entidades almacenadas en un servidor, parecido a un servidor WWW.

El modelo define a una base privada como un tipo especial de nodo de contexto que agrupa nodos juntos, que son los nodos de contexto del usuario y las bases privadas. La base privada puede contener más de una base privada.

Tal que:

- Una base privada podría pertenecer a más de una base privada. Si un nodo compuesto N es contenido en una base privada PB, estos componentes pueden ser contenidos en PB o en la hiperbase pública o en cualquier base privada, de una base privada anidada por otra base privada; y si una liga es contenida en una base privada, estas fuentes básicas finales podrían ser una anotación del nodo.

Intituivamente una base privada es una colección de entidades usadas durante una sesión de trabajo por un usuario.

2.4 MODELO WEBDAV[7]

En la edición colaborativa se presentan problemas en Internet con la redacción de los documentos en un ambiente colaborativo, debido a que un autor deja un texto para que los otros autores puedan anotar algo y estos lo mandan por Internet, el problema se ocasiona cuando se atasca el sistema de correos del autor – jefe, ocasionando que no se pueda ver y ni leer sus correos y por consiguiente no puede revisar las notaciones que hicieron sobre el documento otros autores del mismo equipo, esto ocasiona una perdida de tiempo y de dinero.

El modelo WebDAV es el más conocido y ocupado por los investigadores para trabajar con la edición colaborativa y versiones. Trabajando en grupos el WebDAV es una extensión del protocolo http (Protocolo de Transferencia de Hiper Texto) que provee una infraestructura estándar para la autoridad colaborativa asíncrona (desfasada del tiempo). El WebDAV soporta el uso del http para inter-operar publicaciones de una variedad de contenidos, ofreciendo una interfase común para muchos tipos de repositorios y elaborando una analogía en la red, y la accesibilidad de los sistemas de archivos. Actualmente, el WebDAV trabajando en grupos está definido como un conjunto de extensiones para las bases del protocolo de transferencia de hipertexto (http).

Mientras tanto, con cada revisión consecutiva el WebDAV provee una imagen clara del mejor camino. En vez de pasar documentos de aquí para allá por medio de e-mail, los edita en un sitio del URL. En vez de "pasando la estafeta," un mecanismo de cerrado impide sobreescribir la vías de acceso de los URL, el documento es siempre accesible por medio de su URL, no hay pérdida del tiempo debido a demoras de los correos electrónicos, y los otros colaboradores pueden inspeccionar el progreso como se ha estado realizando. Cuando escribimos las especificaciones del WebDAV, estas fueron necesarias claramente.

El manejo de versiones de WebDAV especifica un conjunto de mecanismos que pueden ser explotados para establecer un conjunto de políticas que permitan a las aplicaciones de los clientes y usuarios encontrar un balance apropiado entre sus necesidades.

WebDAV provee muchos beneficios como [Whitehead,98]:

- Permite que todas las publicaciones puedan ser contenidas en el Web. Ahora los grupos y los individuos pueden usar el protocolo http para publicar directamente sus trabajos.
- Los trabajos en grupos pueden colaborativamente crear documentos en algún sobre usando cerraduras para prevenir conflictos de sobre escritura. Debido a la naturaleza distribuida en como se comporta la red, los grupos de trabajo pueden tener miembros que no sean de la misma organización.

[7] WebDAV es el acrónimo de World Wide Web Distributed Authoring and Versioning, en español WWW con Autoridad Distribuida y Control de Versiones

- WebDAV no contempla restricciones en los tipos de documentos que pueden ser diseñados. El WebDAV provee para el soporte de autoridades con el HTML[8] y XML[9], también puede soportar gráficos, hojas de cálculo, procesadores de palabras y otros tipos de formatos.
- WebDAV y http proveen de una interfase común de un rango amplio para su repositorio, además de un administrador de documentos, un administrador de configuraciones, un sistema de archivos, bases de datos.
- WebDAV permite incorporar controladores para la seguridad de la escritura en la Red. Mas específicamente, el WebDAV define un conjunto de extensiones para la base del http para la siguientes capacidades:
    - **Prevención en la sobre-escritura:** Permite que más de una persona pueda trabajar un documento en cualquier parte, al mismo tiempo. Esta prevención permite que los autores puedan observar las modificaciones que se hallan hecho en el mismo día por otro autor.
    - **Propiedades:** Puede crear, remover y dejar en una cola de espera la información acerca de las páginas, tal como el autor lo hizo, dejando únicamente la última modificación. También incluye la habilidad de marcar ligas de hipertexto entre páginas de algún tipo de recurso.
    - **Administrar los espacios de los nombres:** Crea, remueve y mantienen automáticamente la consistencia de los recursos. Teniendo la habilidad de copiar y mover paginas Web y recibir un listado de los recursos en una colección, similar a un listado de un directorio.

Actualmente, el WebDAV incorpora [Whitehead&Meredith,98]:

- **Administrador de Versiones:** La habilidad de guardar las revisiones importantes para luego revisarlas nuevamente. El manejador de versiones puede trabajar en un ambiente colaborativo que permite que dos o más autores trabajen en el mismo documento en procesos paralelos. Las versiones son guardadas automáticamente con cada modificación que se halla efectuado en el documento original. Apoya el almacenaje de cambios importantes de documento para luego recuperarla.
- **Colecciones Avanzadas:** Parecido a una liga simbólica en un sistema de archivo, la capacidad para agregar una referencia del miembro para una colección que puede apuntar a un recurso de la Red. Además, al ordenar la colección permite a un cliente que especifique el orden persistente de sus recursos dentro de la colección.
- **Control de Accesos:** El WebDAV tienen la capacidad para limitar los derechos de acceso de una persona identificada como el jefe sobre un recurso determinado. WebDAV presume de la existencia de una tecnología de autenticidad fuerte.

[8] HTML acrónimo de HyperText Marked Language – Lenguaje de marcas de hipertextos.
[9] XML acrónimo de X Mark Language – Lenguaje de marcado X

El WebDAV da reglas semánticas de control de versiones [Amsdem&Clemm,99]:

- Un recurso versionado es una abstracción de un recurso el cual está sujeto a control de versiones.
- Una revisión es una versión particular de un recurso versionado. Una revisión inmutable es una revisión que una vez creada, nunca puede ser cambiada sin crear una nueva revisión.
- Una revisión mutable es una revisión que puede cambiar sin crear una nueva versión.
- Una revisión inicial es la primera revisión de un recurso versionado y no tiene predecesores dentro del recurso versionado.
- Las revisiones tienen: un nombre de revisión, un identificador de revisión y una etiqueta.
- Un predecesor de una revisión es una revisión de la cuál está revisión es creada.
- Un sucesor de una revisión es una revisión derivada de esta revisión. Una revisión puede tener un predecesor y múltiples sucesores. La relación es derivado de entre revisiones de un recurso versionado forman un árbol.
- Una historia de revisiones es una representación concreta de los elementos de un recurso versionado incluyendo todas las relaciones de predecesores y sucesores, nombres de revisión, etc.
- Una línea de descendentes es una secuencia de revisiones conectadas por relaciones sucesor / predecesor desde la revisión inicial a una revisión específica.
- Una actividad es un recurso que se refiere un conjunto de revisiones nombradas que corresponden a alguna unidad de trabajo o cambio conceptual.
- Un espacio de trabajo es un recurso que es usado para determinar que la revisión de un recurso versionado debe ser accesado cuando el recurso es referenciado sin un nombre de revisión particular.
- Una regla de selección de revisión específica que revisión de un recurso versionado debe seleccionarse.
- Una configuración es un conjunto de recursos relacionados nombrados donde cada miembro se refiere a una revisión específica de un recurso versionado.

El WebDAV contiene un conjunto de aspectos que puede usarse en una amplia variedad de ambientes que soporten trabajos colaborativos sobre autoridades remotas de documentos. Estos aspectos pueden ser partidos en tres de grupos: prevención de sobre – escritura, propiedades (metadatos) y la administración del espacio de los nombres. A continuación se explicara mejor cada una de los grupos anteriormente mencionados:

2.4.1 PREVENCIÓN DE LA SOBRE – ESCRITURA

Cuando se trabaja en un ambiente colaborativo, los grupos de personas están conformados por más de dos personas, esto puede ocasionar que se pierda el control del mismo, si todos escribieran al mismo tiempo, las actualizaciones que se efectúen en el documento cuando no es versionado se podrían perder sin previo aviso a las demás personas.

Hay diferentes técnicas que se usan para poder solucionar las actualizaciones perdidas.

Las más comunes son [Whitehead,98]:

- **POTS (servicio de teléfono viejo claro):** Este un esquema de una convención social que el grado de la colaboración depende de la comunicación verbal, cuando un autor terminó de trabajar, es seguro para que otro inicie. E-mail puede también ser usado para implementar este plan de control, como puede un objeto físico (batuta) que se pasa de autor a autor en ambientes donde los autores están en diferentes lugares.

- **Cerraduras compartidas**: En este plan, un autor indica a la computadora que controla el acceso al documento que va a ser modificado y la computadora registra el autor que editó el documento. Si otro autor trata de indicar un intento de edición, la computadora anunciará que el documento ha sido editado. Sin embargo, si el segundo autor todavía desea editar, puede hacerlo, presumiblemente por llamar otro autor para negociar acceso y así poder modificar el documento.

- **Cerraduras privadas:** En este plan, un autor indica a la computadora que controla acceso al documento, que él va a modificarlo y la computadora responderá cerrando el documento. Una vez el documento se cierra, no puede ser modificado por nadie a excepción del propietario de la cerradura. Los otros autores que intenten editar el texto tendrán prohibido la acción de modificar por que ellos no poseen la cerradura.

Actualmente, el enfoque del WebDAV es de proveer instalaciones para cerraduras compartidas y privadas. Las cerraduras privadas proveen una garantía más estricta de prevención de conflicto para colaboradores conscientes o durante períodos de contienda alta para un documento

Cerrando comúnmente viene junto con la capacidad de notificar el suceso, para que los otros colaboradores puedan ser automáticamente informados por el sistema cuando una cerradura se ha liberada. Las notificaciones son un mecanismo importante por que colaboradores llegan a ser consciente de las actividades que se están efectuando y puede ocurrir en múltiples niveles de granularidad.

2.4.2 LAS PROPIEDADES DEL WebDAV

La información dentro de la red está compuesta de pedazos de información. Tales como el título, tema, creador, editor, fecha, etc. Esta información es la conocida como metadato y en el WebDAV como propiedades puede usarse para buscar recursos de la red, imponer propiedades o proveer información bibliográfica. Las propiedades son particularmente útiles en buscar recursos de la red debido a las insuficiencias de los índices existentes.

Desde los otros grupos han enfocado en creciente conjunto de metadatos, el grupo de WebDAV se a dedicado a producir aplicaciones para crear, modificar, borrar y recobrar metadatos, estas aplicaciones permiten manipulaciones desde esquemas múltiples, permitiendo el esquema a sí mismo variar con la heredad de uso.

En el WebDAV las propiedades están formadas por el nombre - es un Recurso de Identificación Uniforme (URI), tal como un URL, y el valor es un bien – lo forma la sucesión de Idioma Extensible de Encarecimiento (XML) contenido.

2.4.3 ADMINISTRADOR DEL ESPACIO DEL NOMBRE

En la corriente, publicar / examinar el modelo de la red, hay escasa necesidad que un usuario duplique o renombrar recursos de la red. Sin embargo, la red es usada para la autoridad distribuida, la necesidad que estas capacidades, más la capacidad de que consigan un listado de un directorio, llega a ser sumamente importante. Siendo capaz de descubrir que los recursos actualmente pueblan una porción del espacio de nombre de un servidor de red y la capacidad para copiar, mover y borrar estos recursos, forman junto los elementos claves para administrar el espacio del nombre en una red.

Hay varias justificaciones para agregar capacidad de movimiento y copia. Un recurso puede necesitar ser copiar debido a la titularidad cambiante, con anterioridad a modificaciones importantes, o cuando haciendo un respaldo. Es frecuentemente necesario mover un recurso, por ejemplo debido a la adopción de una nueva convención de nombramiento, o si un error de mecanografía hecho originalmente entraba en el nombre.

La copia y el movimiento tienen ramificaciones con él respecto a propiedades, parecería que todas las propiedades sobre el recurso duplicado o movido deberían ser idénticas a las propiedades sobre el original. Sin embargo, hay realmente dos clases de propiedades: vivo y estático. Las propiedades estáticas tienen la calidad que su valor, una vez conjunto, permanece él mismo hasta que un cliente explícitamente modificarlo. Las propiedades vivas, en el contraste, tienen su sintaxis y la semántica impuesta por el servidor y puede variar en cualquier tiempo. Un ejemplo de una propiedad viva es la longitud contenida de un recurso - cada vez el recurso se actualiza, el valor de la propiedad también se actualizará. WebDAV intentó resolver conflictos entre las propiedades existentes de un recurso siendo movidos y esos que pueden ser impuestos por el servidor o el directorio en que está ser ubicado.

El listado de los contenidos de una colección, una operación parecido a enumerar un directorio de un sistema de archivo, se realiza usando el mecanismo de recuperación de propiedad. La mayoría de las operaciones existentes de listado directivo (tales como "ls" o "dir") proveen el nombre de un archivo y una opción para recobrar conjuntos limitados de propiedades sobre el archivo, tal como su tamaño, el propietario y accesa permisos. En WebDAV la recuperación de propiedad permite una operación única, una recuperación jerárquica de propiedades sobre una colección, volviendo para cada recurso su nombre y pidiendo propiedades.

El control de versiones en el Web ofrece una variedad de beneficios como:

- Provee de una infraestructura para el manejo eficiente de los sitios Web. Los manejadores actuales son construidos con algunas formas de repositorio para que puedan grabar el historial de las revisiones de los recursos individuales.

Básicamente la capacidad de controlar las versiones son requeridas para soportar al sistema.

- Permite el desarrollo paralelo y la actualización de un recurso. El registro del sistema de versiones cambia por la creación de un nuevo objeto.
- Provee de una estructura para coordinar los cambios de los recursos.
- Permite la búsqueda a través de versiones de los recursos pasados o alternativos.
- Provee nombres estables que puedan soportar los vínculos acumulados por las anotaciones y los soportes de los vínculos del servidor.
- Permite la representación explícita de la semántica de un recurso con múltiples estados. El sistema de versiones representa que los recursos que tienen un historial explícito y una identidad persistente cambien sus estados en el transcurso de la historia de un recurso.

Como se observa el WebDAV es el método que se preocupa más de los recursos que se tienen para poder manejar el control de versiones.

C A P Í T U L O 3

Sistemas que implementan el control de versiones

En el campo de la edición colaborativa, los investigadores se han preocupado por poder solucionar el problema del control de versiones dentro de los documentos colaborativos. Además de cómo poder manipular las versiones de los documentos.

Un gran problema al enfrentamos fue que muchas de las aplicaciones sólo son prototipos de laboratorio y sólo una de esas aplicaciones maneja lo que son los modelos vistos en el capítulo anterior.

3.1 SISTEMA MUCH

El sistema MUCH fue desarrollado por la Universidad de Liverpool, en el Departamento de Ciencias de la Computación. La visión de crear el Sistema MUCH fue de tener un sistema que permita a un grupo de personas trabajar colaborativamente en un ambiente de hipertexto e hipermedia. Con esto, permitirá que las computadoras y las redes de telecomunicaciones fueran más amplias, para que muchas personas accesaran a la información a través de pantallas [Rada,96].

La última versión del sistema MUCH (en total tiene 6 versiones) contiene una interfaz de multimedia Andrew y está basada en su propio sistema manejador de bases de datos. Las anteriores versiones del sistema fueron utilizadas para diferentes proyectos como: la versión 5 en el proyecto CEC Delta Practitioner (es un software de reuso en 1991) y en el CEC Delta Oscar (proyecto de colaboración de autoridad en 1992).

La última versión ha sido usada por un equipo de 20 personas en la creación de un libro, artículos y propuestas fundamentadas.

El diseño del sistema MUCH fue realizado por el investigador Roy Rada[10]. La arquitectura del sistema MUCH se estructura por una red semántica, una ingeniería transversal dinámica y una interfase jerárquica de la red semántica generada por el programa transversal. En términos del modelo de referencia de hipertexto Dexter, la red semántica en el sistema MUCH, corresponde a la organización de la capa de almacenaje y a la interfaz jerárquica que ayuda al usuario en la capa de presentación para la navegación a través de la red semántica [Rada&Chen,96].

El programa transversal usa la combinación del algoritmo de búsqueda depth-first y breath-first. La combinación transversal, primero colecciona todos los nodos desde su vínculo y entonces escoge el nuevo nodo por el principio depth-first. El sistema permite a los

[10] Investigador de la Universidad de Liverpool en el Dpto. de Ciencias de la Computación

usuarios alterar la ruta transversal ya predispuesta para configurar un número de parámetros opcionales.

El sistema MUCH [Rada&Chen,96] fue diseñado para soportar la escritura colaborativa asincrona. El sistema tiene una arquitectura cliente - servidor. El servidor del sistema MUCH, llamado monitor, maneja las transacciones concernientes al control de concurrencia y varias facilidades colaborativa. Un modelo teórico en el sistema MUCH ha sido desarrollado por la escritura colaborativa y un número de estudios empíricos han sido conducido para observar los efectos que envuelve la colaboración de hipertexto.

Como es un sistema que entra dentro del área CSCW, es muy importante el manejo en el aspecto social sobre las conductas de los integrantes de los grupos de personas que colaboran en este trabajo. La estructura social introducida por el sistema MUCH es construida en las siguientes acciones, la cual comparte el camino de la interacción del usuario con una base de datos de hipertexto colaborativa:

- La función UNFOLD expande la vista jerárquica en breadth-first. El uso de la función UNFOLD recupera y muestra todos los nodos hijos del nodo focal).
- La función READ para cargar y mostrar el contenido de un nodo en la vista jerárquica.
- La función LOCK desbloquea el contenido del nodo focal desde la edición concurrente, pero otros usuarios aún puedan usar la función READ en el nodo.
- La función INFO para mostrar la información almacenada en un objeto incrustado de cada nodo.
- La función UPDATE salva los cambios desde la última actualización o desde la última desbloqueada y lo realiza la actualización contenida por un usuario concurrente.

En cada nodo, en el sistema MUCH contiene un pedazo de información de multimedia. El sistema persistentemente almacena la evolución de la información de cada nodo con su vinculación. El mantenimiento de su información es accesible por la función INFO, la cual está disponible desde cada nodo en la red semántica. Un usuario puede verificar la historia de los nodos, mostrando la información tal como quien modifico recientemente el nodo y cuántas veces ha sido visitado ese nodo por otros usuarios diferentes del creador del nodo.

En la figura 7 se muestra el cuadro de diálogo de la función de INFO. El número 2 en el campo de selección de créditos indica que el nodo ha sido visitado dos veces por otros usuarios que no son el creador, quien tiene el nombre del usuario *much* y el contenido de este nodo no ha cambio desde 14 de julio de 1994 [Rada&Chen,96].

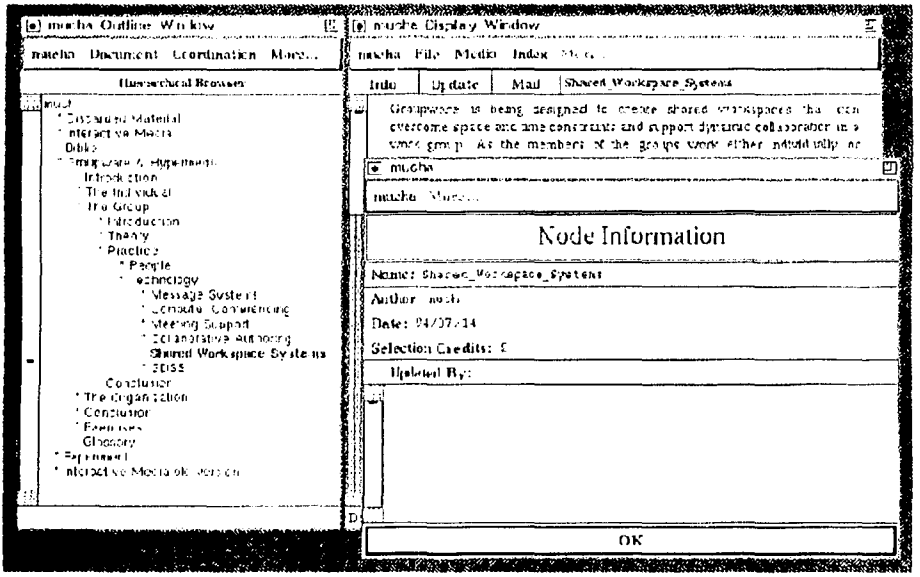


Figura 8 Cuadro de dialogo de la función INFO

La anterior figura 8 muestra la pantalla principal del sistema MUCH.

Recorriendo el patrón de conducta con la función INFO se revelan algunas señales dentro del uso natural de esta función en la escritura colaborativa asincrona. Además ayuda al diseño del sistema para capturar lo más sobresaliente del contexto de conducta en las relaciones dinámicas.

Se realizó un estudio [Rada&Chen,96] para observar cuantas personas entran a trabajar con la función INFO. En el estudio se utilizaron 5 bases de datos las cuales estaban divididas de la siguiente manera: 1 para investigadores y las restantes 4 para un grupo de graduados de la Universidad de Liverpool. El estudio se realizó en un periodo de 3 meses. El grupo de investigadores era de 15 personas y de los graduados era de 30 personas.

Para los alumnos se realizó el estudio en dos fases. En la primera uso el cuadro de texto electrónico en el salón de clases con una duración de 3 semanas, en la segundo fase formaron 8 grupos de 3 integrantes para escribir. A los estudiantes se les permitió usar las herramientas de comunicación externa.

En el estudio se utilizó la distribución de Poisson para poder manejar los datos y sacar un análisis. Los autores utilizaron la distribución con la finalidad de manejar intervalos de tiempo.

La distribución Poisson nos dice que [D'Cantu,80]:

Si X es el número de ocurrencias de un evento aleatorio con un intervalo de tiempo o espacio, la probabilidad de que X ocurra, está dada por la función de probabilidad de Poisson:

$F(x) = (e^{-\lambda}\lambda^{x}) / x!,\ x= 0,1,2,.....$

- La letra griega λ (lambda) es el parámetro de esta distribución y el *promedió* de ocurrencias del evento aleatorio en el intervalo.
- El símbolo e es una constante cuyo valor aproximado a 4 cifras decimales es 2.7813
- Un experimento que cumple con las siguientes condiciones se llama experimento de Poisson.
- Los eventos ocurren en forma independiente, es decir, las ocurrencias de un evento en un intervalo de tiempo o espacio no afecta la probabilidad de una segunda ocurrencia del evento en el mismo u otro intervalo.
- Teóricamente es posible que el evento pueda ocurrir infinitas veces en el intervalo.
- La probabilidad de que ocurra un evento en un intervalo es proporcional a la longitud del intervalo.

En el estudio los datos se obtuvieron por dos fuentes:

- Acciones ejecutadas por los usuarios en la capa de presentación.
- Una representación estructural en la red semántica en la capa de almacenaje.

Cada acción en la capa de presentación se registra en términos de cuando es tomada quién activó la acción y dónde se activó en el espacio de información de hipertexto. La representación estructurada de la red semántica fue usada para hacer juego con una acción INFO con el nodo donde la acción es activada. La acción INFO permite al usuario ver la información grabada en el objeto INFO de las visitas ocurrentes del nodo.

Este proceso de interactividad con la base de datos compartido puede ser caracterizada por una secuencia de eventos. Cada evento toma el lugar en asociación con un nodo particular. Cada nodo tiene un conjunto de atributos de la evolución, tal como, en qué posición jerárquica se encuentra la capa de presentación y una lista de usuarios quienes han modificado el nodo recientemente. El resultado es una subsecuencia de un conjunto de

atributos de cada intervalo de tiempo. Por ejemplo, ponemos n.attr denota el valor del atributo attr del nodo n y la secuencia del los nodos $\{n_{i1},n_{i2},..,n_{ik}\}$ corresponde a la ocurrencia de los eventos en i-intervalos en la subsecuencia, la siguiente ecuación define la serie de tiempo de una conjunto de variables basadas en el atributo original level:

$s_i(level) = _{j=1,2,...,k} n_{ij}.level$

Un número de un nuevo tiempo son derivados por un conjunto de observaciones en los intervalos de tiempo sobre la relación de los valores. Estos tiempos son coleccionados llamados rutas de vistas inconscientes en la base de datos usada. Esta serie de tiempo son estándares nuevos dentro del proceso.

Por ejemplo:

$Zlevel_i = s_i(level) / s.d.$

Donde:

s.d. es la derivación estándar de la serie de tiempo $\{s_i(level)\}$. El resultado de la serie de tiempo Zlevel = $\{Zlevel_i\}$ indicando el promedio de la posición de estos nodos en la vista jerárquica en el cual la acción INFO ha sido usada frecuentemente. Varias series de tiempo en este estudio son generadas similarmente. Ver la siguiente tabla 1 donde se nota algunas series de tiempo que son consideras en el análisis preliminar:

Variable	Definición	Nota
Hour	Una serie de tiempo particular impuesta en Windows.	S/A*
Phase	Un paso de la función para actividades de lectura y escritura.	S/A
Zcredit	Acumulación de números de visitas por otros usuarios que ellos son autores.	S/A
Zlevel	Niveles de los nodos relacionados con el evento de INFO.	S/A
Zlink	Total de números de ligas salidas de los nodos envueltos.	S/A
Zlink	El tamaño de los nodos envueltos (contador de ligas).	S/A
Zsize	El tamaño del grupo de usuarios ocurrentes.	S/A
Zword	El tamaño de los nodos envueltos (contador de palabras).	S/A

Tabla 1 Muestra las series de tiempo que son consideradas en el análisis preliminar.
S: Estandarizados A: Conjuntos

En el análisis de datos del evento INFO fueron medidas en números enteros no negativos. Los datos discretos, tales como, el contador de eventos no puede ser analizado con el

análisis de regresión clásica porque el método requiere una variable dependiente continua. En un análisis preliminar se escogió trabajar con la distribución de Poisson, un modelo de regresión de Poisson fue usado para capturar las relaciones entre las ocurrencias de INFO y varios aspectos contextuales de la base de datos de hipertexto colaborativa usadas por los estudiantes.

En el campo de la probabilidad se utilizó a Kolmogorov-Smirnov para un buen manejo estadístico de las ocurrencias de los eventos de INFO con la distribución de Poisson. Un modelo de regresión de Poisson puede ser especificada por un modelo lineal en el cual la variable dependiente sigue una distribución de Poisson. Un procedimiento GENLOG (para generar un modelo de análisis de registros lineales) en el paquete estadístico SPSS fue usado para modelarlo. Las ocurrencias de los eventos de INFO son valores en celdas de la tabla asociada de contingencia con el modelo de registro lineal. Ponemos Y(t) denota las ocurrencias del evento INFO en un intervalo de una hora de tiempo t, $X_i(t)$; el valor de i de la variable explicativa con este coeficiente $_i$ en el mismo tiempo de intervalo t, el modelo especifico tiene la siguiente forma:

$$Ln(Y(t)) = constant + {}_iX_i(t)$$

Sus intervalos de una hora son consecutivos en toda la semana tomando las 24 horas del día y los 7 días de la semana. Las variables explicativas fueron transformadas en sus datos estándares. Un adecuado filtro en el modelo de regresión de Poisson fue seleccionado subsecuentemente para representar el contexto de conducta de las relaciones del nuevo análisis.

Los resultados obtenidos en el manejo del evento de INFO es estudiado por medio de las relaciones entre las ocurrencias de INFO y varios contextos, las variables explicativas en la base de datos colaborativa de hipertexto. Las ocurrencias de INFO son indicadas por un crecimiento concerniente a algo social y de coordinación. El modelo resultante del contexto-conducta es desarrollada dentro de la estructura del estudio, como también provee de evidencias empíricas para la evaluación de modelos existentes del trabajo colaborativo en CSCW.

El uso de las cinco bases de datos en el sistema MUCH resultaron en cientos de transacciones durante los tres meses de duración de la observación. La escala de uso para cada bases de datos es variada. La base de datos que fue accesada por el grupo de usuarios (Usuarios = 30) con el espectro más grande de funciones desde la capa de presentación (Activo = 25). En la tabla 2 muestra todas las escalas de uso para cada base de datos, en términos del número total de transacciones observadas, el número total de nodos únicos incluidos, el tamaño de los equipos de usuarios y el número de funciones distintas que requirieron.

Database	Transaction	Node	User	Action	Started	Ended
Groupware	12089	750	11	22	Tue Apr 19	Mon July 25
Class	9247	474	30	25	Wen Apr 20	Mon July 15
Management	7068	698	16	22	Wen Apr 20	Thu July 21
OSCAR	3148	315	7	18	Tue Apr 19	Mon July 25
General	1590	96	10	21	Tue Apr 19	Fri July 15

Tabla 2 Uso de las 5 base de datos en los 3 meses de observación

El promedio del uso de la función INFO es acerca 1% en las cinco bases de datos. En la siguiente tabla (ver 3) muestra las ocurrencias de los eventos en los intervalos de tiempo.

Ocurrencias	0	1	2	3	4	5	6	7
Intervalos	2269	19	6	2	1	1	3	1

Tabla 3 Muestra las ocurrencias de los eventos INFO

Varios modelos de regresión de Poisson fueron hechos a la medida para los datos de las series de tiempo de las ocurrencias de los eventos de INFO y su correspondiente característica contextual. En particular, el modelo específico como siguen los resultados por medio de una ecuación de prueba de bondad para los datos:

$$\text{Ln(count)} = {}_0 + {}_1 *\text{Zcredit} + {}_2 *\text{Zlevel} + {}_3 *\text{Zlink} + {}_4 *\text{Zword} + {}_5 *\text{Zsize} + {}_6 *\text{hour}$$

Las variables explicatorias, o variables independientes fueron especificadas como variables covariantes en el modelo, excepto la variable de la hora, la cual es categórica.

Parameter	Estimate	SE	Z-value	Marginal
X_k	$_k$		* p< 0.05	effect
Zlevel	19.2786	7.5654	2.55*	0.5938
Hour	.2060	.3023	.68	0.0063
Zword	.9397	9.7437	.10	0.0289
Zlink	-.6285	1.4958	-.42	-0.0194
Zcredit	-11.2120	4.1423	-2.71*	-0.3453
Zsize	-15.2472	2.6603	-5.73	-0.4696
Constant	1.4017	3.1006	.45	0.0436

Tabla 4 Muestra los resultados de las estimaciones por el modelo y las pruebas asociadas significativas.

En la tabla (ver 4) se muestran los resultados del modelo de regresión de Poisson de las relaciones entre los eventos de INFO t los caracteres contextuales del camino de activación en el mismo intervalo de tiempo.

Se utilizó el modelo de regresión de Poisson porque puede también ser interpretado en términos de los efectos marginales de una x en el contador de INFO. El efecto marginal de x_k esperado y es obtenido por $_k$, donde = 0.0308.

El efecto marginal de Zlevel sobre INFO es (0.0308)(19.2786) = 0.5938. en otras palabras, si el valor de Zlevel crece por 1, el esperado del contador de INFO crecería por 0.5938. Al contrario, el efecto marginal esperado del contador de INFO es (0.0308)(-11.2120) = -0.3453, sugerido que el esperado del contador INFO decrecería por 0.3453 como el Zcredit crece por una desviación estándar [Rada&Chen,96].

Si la variable falsa fase es incluida en el modelo para especificar las dos fases de lectura y autoridad, la variable fase tendrá el efecto más predominante en las ocurrencias esperadas de INFO. Si consideramos la fase de escritura entonces crecerá la probabilidad del contador de INFO. La frecuencia del evento de INFO en efecto creció cuando el grupo fue emergiendo en el sistema MUCH.

El modelo de regresión de Poisson sugiere que la función INFO es más frecuentemente necesitado en nodos involucrados en un trabajo real. El uso de la función INFO tiende a ser ejecutado por un problema especifico en un modo de trabajo independiente. El evento INFO ocurrió raramente con todas las bases de datos del sistema MUCH. Las ocurrencias de los eventos fueron puestas en una distribución de Poisson. La relación entre el evento INFO y algunas características contextuales de cómo los usuarios interactuaban con las bases de datos compartidas revelaron algunos modelos relacionados para la interacción entre los usuarios y las bases de datos. Estos factores afectaron la conducta del usuario para usar particularmente la función de INFO [Rada&Chen,96].

En conclusiones de este estudio se reveló que una característica específica de un sistema de base de datos de hipertexto compartido, podría tener impacto en los procesos de la interacción social y en la conducta de los usuarios. En este sentido, todos los sistemas individuales que puedan compartir el proceso de las personas trabajando e inevitablemente la estructura social la cual es implantada en su trabajo.

3.2 SISTEMA CLEARCASE[11]

El sistema fue realizado por Silicón Graphics para el manejo de configuraciones y control de versiones, su autora es Linda Stewart que trabaja en *United States Federal Aviation Administration* [Stewart,94].

La decisión de crear un software para el manejo de versiones, ha presentado varias interrogantes las cuales hoy en día, siguen sin respuesta; como el control de versiones, la comparación de versiones, la capacidad de la información. El volumen de puntos

[11] ClearCase elaborado por Linda Stewart en USA Federal Aviation Administration

importantes en la actualidad no es el problema, pero como el software ha resuelto algunos puntos y la versión primera del software se libera formalmente, el énfasis cambiará para resolver estos puntos. En la actualidad, los cuestionamientos han reasignados a los desarrolladores y a los usuarios (autores) para resolverlos en investigaciones y en desarrollo de herramientas disponibles para proyectos y negocios pequeños que carezcan de personal suficiente para crear sus propios sistemas o herramientas.

El sistema ClearCase fue creado para apoyar el control de versiones. El sistema ClearCase presenta la capacidad de manejar múltiples versiones de software desarrollado. Permite relacionar los documentos y la información de las bases de datos, usando pistas para las versiones en un software en construcción, ejecutando programas individuales o en todas las liberaciones de los acuerdos de la definición de los usuarios para las versiones especificadas y forzando las políticas en los sitios desarrollados específicamente.

El sistema ClearCase es una solución únicamente para las empresas que tengan un poder monetario fuerte, para que pueda trabajar con este sistema, debido que requiere de pasos extras y de tiempo en el flujo de trabajo de una compañía.

Utiliza herramientas relacionadas al sistema ClearCase [Stewart,94]:

- WordPerfect 5.0 para UNIX incluyendo la configuración de macros para la conversión de documentos.
- Configuración de SCRIPT para convertir páginas principales de CGI a un formato HTML
- Formatos de Mosaico para base de datos.

Los problemas obtenidos y resueltos por ClearCase:

- **Indexación:** La indexación parece ser un mayor reto por algún sistema de hipertexto sin importar si es distribuido o abierto. Con la decisión en el manejo de las versiones de la documentación por línea dentro del Web, los problemas que se obtienen de la indexación se expanden desde una simple necesidad efectiva, como la ingeniería de búsqueda y los resultados de los documentos, la necesidad de buscar una versión especifica de un documento actualizado. El sistema ClearCase busca las estructuras de los archivos virtuales, pero están disponibles únicamente en las computadoras donde esté corriendo el sistema.
- **Formas / Escrituras:** Las formas y las escrituras están relacionadas por el documento fuente, también presentan un reto por la necesidad de tener disponible el mantenimiento de sus correspondientes versiones. ClearCase provee esta habilidad pero con la limitante de sólo estar disponible en las computadoras donde esté corriendo el sistema.
- **Anotaciones:** Las anotaciones personales es una gran ayuda para los autores, debido a que son comentarios sobre el documento, son usados ampliamente en el sistema. Las

anotaciones grupales son la gran necesidad en nuestro desarrollo del sistema, esto depende sobre la interfase desarrollador - usuario y las interacciones que tendrá el usuario y el sistema que maneje las versiones. Uno designa la forma de escritura que se usará permitiendo que las anotaciones de grupos que escriben fuera de la dirección regresen para un archivo que se muestra. Una situación real existe cuando uno quiere indicar la posible necesidad de poder acceder varias versiones e anotaciones de un mismo documento, para poder observar los diferentes comentarios que se tienen del documento.

- **La estructura de archivos y las conversiones de nombres:** La estructura de archivos, la conversión de nombres presentan un gran problema cuando al intentar cruzar las plataformas, ejemplo de UNÍX a DOS. Nosotros escogemos extender el nombre para poder trabajar con el cambio de sistema, pero este intento en un sistema distribuido tiene que ser considerado. No es seguro las implicaciones de la conversión de nombres en el futuro de los documentos en línea y sus subsecuentes versiones de cada documento.

- **Factor Humano:** Quizás el factor humano tal como procedimiento y permisor son cuestionados como su relación con las versiones de los sistemas de hipertexto, sin embargo, los sistemas mas poderosos pierden su efectividad si no están proveídos por un plan de trabajo, uso y administrado. El factor humano está adentro de los problemas que se tienen en un sistema de hipertexto, pero lo que se hace en este sistema es que al factor humano se le toma en cuenta para poder resolver el problema, no es la causa pero si es la solución.

- **Mejoras:** La compatibilidad entre los buscadores, editores, etc., parece ser su mayor problema. Es seguro que la aplicación es rica en sus respaldos de información acerca de la compatibilidad. En el presente nosotros mantenemos toda la vieja aplicación usada por las versiones como parte de su liberación, pero esto es un problema para el espacio en disco.

- **Vinculaciones:** Nuestro grupo es atendido por ligas por un simple documento desde guías de pista, manuales de usuario, archivos de ayuda y notas. Una vez obtenido, esto ya no es un problema para las computadoras donde corre el sistema ClearCase. La necesidad para copiar la versión liberada es un estándar de la estructura de archivos duplicados dentro de la WEB.

El sistema ClearCase se ha mejorado para brindar un mejor desempeño para el trabajo colaborativo. A continuación se mostrarán algunas pantallas que tiene la versión 2 de ClearCase [Eaton,94].

En la figura (9) se muestra el inicio de una sección con ClearCase V2. El administrador muestra que uno puede trabajar con versiones anteriores, configurar el sistema y la administración. Despliega la información con dar doble clic en una de las versiones mostradas .

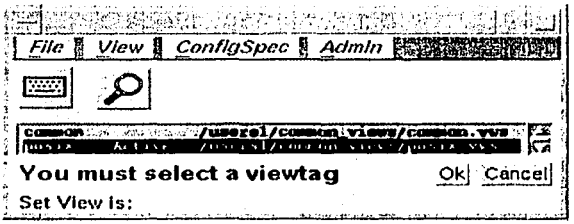


Figura 9 Inicio del ClearCase

En la edición de archivos, el sistema abre un archivo seleccionando al mismo o un elemento por default que toma el editor o uno especificado por los variables usadas por el ambiente del editor. En este cado se muestra un ambiente de escritorio común. Como lo muestra la figura (10).

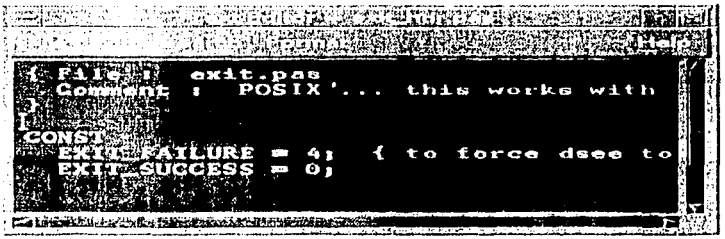


Figura 10 Editor de archivos

Su nuevo administrador de archivos (File Browser) ha tenido una extensión en su lista de funciones. Como lo muestra la figura 11.

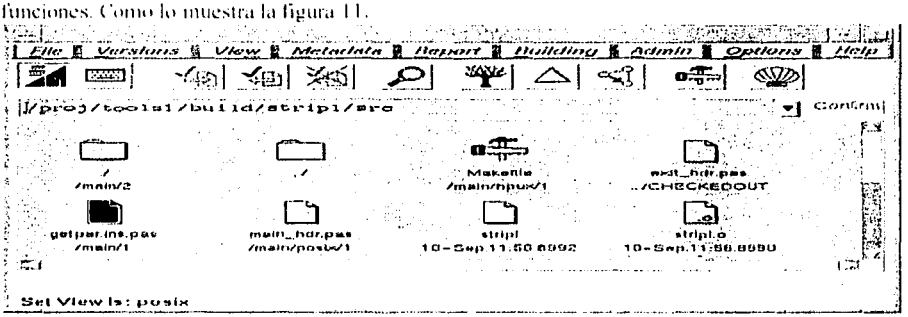


Figura 11 Administrador de archivos

Elementos y archivos podrían ser mostrados como íconos o líneas de texto, según el usuario escoja. El ícono del archivo puede ser modificado como sucede con otros sistemas operativos. Como se muestra en la figura (12).

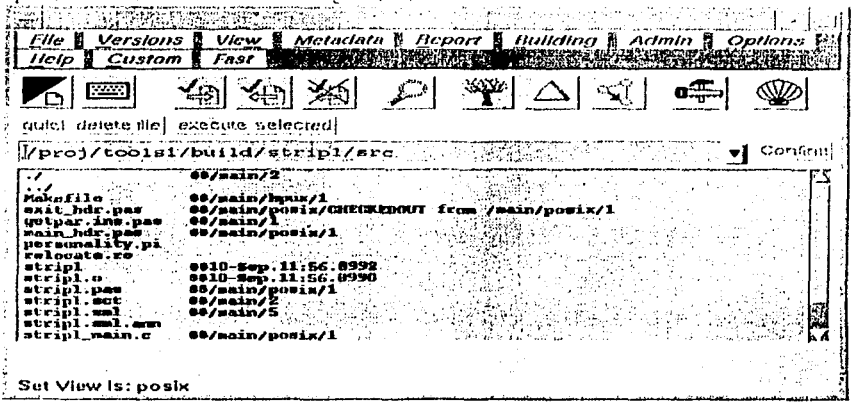


Figura 12 Administrador de archivos en modo texto.

La nueva versión muestra lo fácil que es leer y manipular las versiones. Se selecciona un nodo permitiendo que otras operaciones estén sobre el documento seleccionado. Figura13.

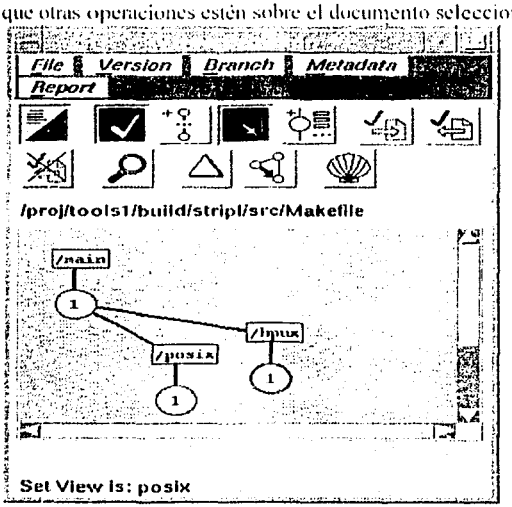


Figura 13 Muestra como se representa un árbol con versiones.

Comando Describe

El comando "*describe*" es una de las nuevas mejorías que se tiene en el sistema ClearCase V2, el cual podría ser activado desde la barra de herramientas por un clic. Muestra la información acerca de una versión, su ruta, creada por que autor. Figura 14.

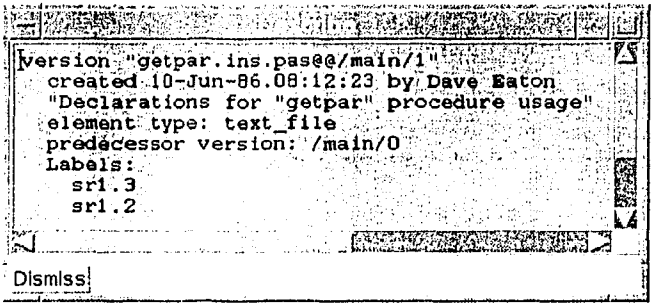


Figura 14 Descriptor

Comparador - Cleardiff

La nueva herramienta compara versiones de un mismo documento y muestra las diferencias usando distintos colores. Como se muestra en la figura 15.

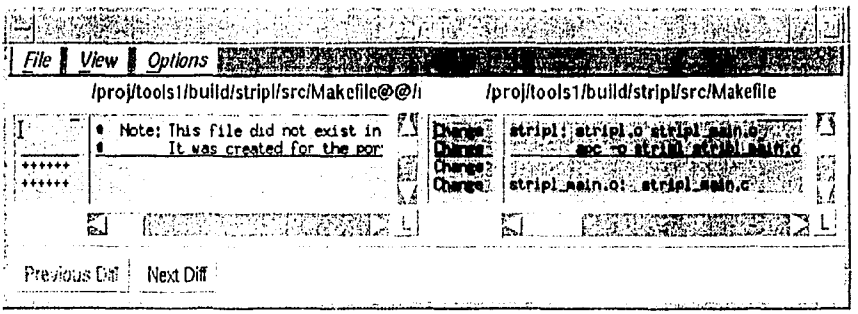


Figura 15 Comparador

En sí, las mejoras han sido en cuestión del manejo de los archivos y de su presentación, el manejo del árbol de versiones que ahora es más grande para poder visualizar todo su árbol jerárquico.

3.3 SISTEMA CVS[12]

El sistema CVS es un sistema de control de versiones. Usando este sistema, nosotros podemos registrar en un historial las modificaciones que se han hecho al documento fuente. Los errores son arrastrados por culpa de una modificación y al no poder detectarlos, se detecta el error hasta un largo tiempo después de que se realizo la modificación, con el sistema se puede fácilmente recobrar la última versión que se vio exactamente cuando cambió por causa de un error [Peña,98].

CVS también ayudará si eres parte de un grupo de personas que trabajan en un mismo proyecto. El sistema hace sencillo la parte de la sobre – escritura cuando se hacen los cambios pequeños que son los de más cuidado. Algunos editores, gustan del GNU Emacs, trata de dar seguridad que el mismo archivo no sea modificado por dos personas en el mismo tiempo. Desafortunadamente, si alguien utiliza otro editor, el mismo editor del CVS no le permite trabajar con el documento. El sistema CVS resuelve este problema aislando al diferente desarrollador en su editor. Todos los desarrolladores trabajan en su propio directorio y CVS combina los trabajos de cada desarrollador cuando se necesite hacer.

El CVS incorpora módulos o estructuras jerárquicas de archivos para el manejo de las versiones.

El control de versiones es una especie de tipos de software usadas para el manejo de cambios y registros. Se utiliza el sistema de control de versiones para:

- **Mantener una base de datos con todos los documentos:** Esto ocasiona que se lleve un mejor manejo de los historiales de los documentos y del autor que hizo la modificación.

- **Gestiona las peticiones de E / S de la base de datos:** El sistema se encarga de llevar una administración de las peticiones que hacen los usuarios (autores), para pedir una información de los documentos, sacando una versión, o el documento original.

- **Mantiene una historia de la base de datos:**

    - **Registro de accesos:** Registra los accesos que se han efectuado dentro de la base de datos por los autores.

    - **Registro de modificaciones:** En su historial, el sistema de control de versiones se encarga de administrar las versiones que se han hecho de un documento original.

- **Permite recuperar documentos previos a cambios:** Su sistema nos sirve para poder recuperar versiones anteriores de un documento.

[12] CVS es el acrónimo de Control Versión System en español Sistema de Control de Versiones

- **Permite diseñar distintas ramas de versiones:** Como se crea un grafo, este tiene ramificaciones de los cambios que se hicieron al documento, o se puede hacer por medio de los autores.

El sistema CVS esta basado en RCS[13] (Conjunto de utilidades de bajo nivel con las cuales se estructura CVS). RCS [Lee,97] comprende un conjunto de programas diseñados para guardar el registro del cambio individual de un archivo. Un objetivo de CVS es proveer un conjunto de funciones que permitan el manejo de una colección de archivos RCS como un solo objeto. CVS almacena todas las versiones de un documento en una manera inteligente, que solamente guarda las diferencias entre versiones. CVS también ayuda si el autor es la parte de un grupo de gente que trabaja sobre el mismo proyecto. Asegura que no se modifique dos veces el mismo documento. Su seguridad la hace aislando a los diferentes autores unos de otros para que no se afecte el documento y no haya conflictos entre los autores.

En el estudio se definen conceptos que muchas veces se confunden con el sistema CVS de lo que puede hacer [Signum,98]:

- **CVS no es un constructor de sistemas:** Aunque su estructura de sus repositorios y sus módulos de archivos interactúan con su constructor de sistema (Makefile), ellos son esencialmente independientes. CVS no es un constructor de sistemas como tal, sólo almacena los archivos en una estructura de árbol para luego recuperar los archivos.

El CVS no especifica como usar el espacio del disco en trabajo de revisión de los directorios. Si uno escribe en su MAKEFILE o en escritos en cada directorio para que ellos tengan el conocimiento de la posición relativa de todas la cosas.

- **El CVS no es sustituido por un manejador:** Sus manejadores y los lideres de proyectos son expertos para hablar frecuentemente de hacer seguro que los calendarios, los puntos de trabajo se cumplan y un buen manejo de los nombres de las versiones dentro del sistema. El sistema es un instrumento para saber cual es el documento fuente en turno.

- **El CVS no es un sustituto para el desarrollo de la comunicación:** Cuando se encara un conflicto dentro de un archivo único, la mayoría de los desarrolladores se administran para resolver sin tener que esforzarse demasiado. Pero una definición más general de conflicto incluye problemas demasiados difíciles de resolver sin la comunicación entre desarrolladores. El CVS no puede determinar cuando cambia simultáneamente una versión sin un archivo único o a través de la totalidad de la colección de archivos, que lógicamente el conflicto podría ser en cualquier otro archivo.

[13] RCS es el acrónimo de Revisión Control System en español- Sistema de Control de Revisión

- **El CVS no tiene el control de cambios:** El control de cambios se refiere para el número de cosas que pueden ser cambiadas de un documento original. El primero de estos, puede significar "la pista del error", que es el inicio de reportar los errores paso por paso en la base de datos y el estatus de cada uno de los errores. Otro aspecto del control de cambios es el guardado de la pista del factor que cambio en varios archivos. Además de los aspectos mencionados, el último es la habilidad de guardar la pista del estatus de cada cambio. Algunos cambios tienen que ser escritos por un desarrollador, otros tienes que ser revisados por un segundo desarrollador. En general, el camino que utiliza el CVS es generar un archivo que muestre la diferencia que existe en las diferentes versiones y luego manda un correo a alguien que pueda aplicar el archivo usando una utilería de CVS.

- **El CVS no es un programa de examen automático:** No revisa las versiones que se han creado automáticamente para ver si lo que se creó está bien o fue por un error del mismo sistema, por un desarrollador o autor.

- **El sistema no tiene un modelo para incorporar procesos:** Algunos sistemas proveen de un camino para asegurar que los cambios sean implantados mediante diversos pasos, con diversas aprobaciones que se necesitan. Generalmente, uno puede acoplar estos con el CVS pero esto podría ser más trabajo. En estos casos si se quisiera usar algunas instrucciones de CVS para el manejo de archivos, se requeriría que los pasos fueran ejecutados antes que el CVS permita una revisión. También se considera si se caracteriza como ramificaciones y etiquetas que pueden usarse para desempeñar tareas como hacer el trabajo en un desarrollo de un árbol y entonces combinar cambios seguros sobre el árbol estable, únicamente probado por el sistema.

A diferencia de los sistemas existentes de edición colaborativa, el CVS ofrece muchas ventajas para la edición colaborativa. Las ventajas que ofrece son:

- CVS es un software gratis (gracias al GNU).
- Soporta varias plataformas de trabajo, como Mac, Win y UNIX.
- Al momento que se baja el sistema, CVS es entregado con los fuentes completos
- CVS adiciona la noción de módulos o colección jerárquica de archivos

Como se ve la ventaja de que se te entregue los códigos fuentes de un sistema da la oportunidad de explotar mejor al mismo sistema.

Algunos comandos que se utilizan en el CVS[Peña,98]:

cvs checkout (o cvs co)

Para crear una copia local de los ficheros de un directorio específico se ejecutará 'cvs checkout module'.

cvs update

Actualiza su copia de un directorio específico con los cambios que se hayan producido en la copia del repositorio (repository) ejecutando 'cvs update'. Le mostrará los ficheros actualizados precedidos por ú' y cuales han sido modificados por ti y no han sido actualizados 'M'.

Los ficheros que contienen 'choques' están precedidos por 'C'. Dentro de los ficheros, los 'choques' están marcados de la forma '<<<<' y '>>>>'.

cvs commit

Cuando piense que sus ficheros están listos para introducirlos en el repositorio (repository) para que el resto de los desarrolladores lo vean, ejecute 'cvs commit'.

cvs add y cvs remove

Puede que los cambios que haya efectuado impliquen un nuevo fichero o eliminar uno ya existente. Las instrucciones a utilizar son: cvs add 'fichero' cvs remove 'fichero' Puede hacer una actualización de los ficheros centrales después de ejecutar las instrucciones de adición y borrado para que surta efecto.

cvs release

Cuando haya terminado con su copia local de ficheros por ahora y quiera borrarla use 'cvs release module'. Primero debería hacer una actualización del directorio en el repositorio (repository).

Si desea que CVS también elimine el subdirectorio (del directorio de trabajo o module) y su copia local de los ficheros entonces ejecute 'cvs release -d module'.

cvs log

Para mirar los mensajes de actualización de los ficheros utilice 'cvs log fichero(s)'.

cvs diff

Para ver las diferencias entre su versión de los ficheros y la versión del repositorio (repository) haga 'cvs diff fichero(s)'

cvs tag

Una de las características interesantes de CVS es la posibilidad de marcar todos los ficheros en un directorio específico (module) inmediatamente con un nombre simbólico. Utilice: cvs tag nombre_simbolico fichero cvs co -r nombre_simbolico module

cvs rtag

Al igual que 'cvs tag', rtag marca las versiones actuales de los ficheros, no lo hace en sus copias locales pero sí en los ficheros del repositorio (repository).

Para etiquetar todas las librerías cvs rtag LIBRARY_2_0 lib

Esto etiqueta todas las librerías de los directorios del repositorio (repository) que utilicen 'lib' con LIBRARY_2_0.

cvs history

Para informarse sobre sus repositorios de CVS (CVS repository) utilice la instrucción 'cvs history'.

cvs history -a -o
cvs history -a -T
cvs history -a -e

3.4 SISTEMA HYPERPROP

Basándose en el modelo de Contexto Anidado se crea el sistema HyperProp [Soares,98].

El sistema HyperProp para ambientes de autoridad, el cual se basa en tres diferentes vistas como lo muestra la siguiente figura 16.

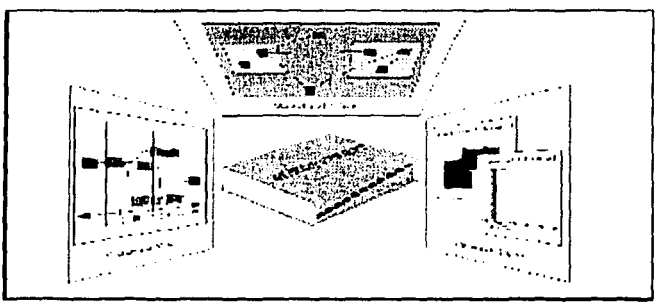


Figura 16 Muestra las diferentes vistas de un documento de HyperProp.

El sistema HyperProp esta constituido por diferentes vistas que son:

- La primera vista, llamada Vista Estructurada, soporta búsquedas y ediciones de una estructura lógica de un hiperdocumento, proporcionando características para la edición de nodos, ligas y grupos dentro de esta composición, como lo muestra la

figura en la parte superior. En esta vista los nodos son representados por rectángulos, las ligas son representadas por líneas y su contenido de las relaciones de composición de nodos son representados por la inclusión de rectángulos y líneas. Fue desarrollado un sofisticado algoritmo de filtro para evitar a los autores que se desorienten principalmente en documentos complejos con nodos y ligas.

- La segunda vista, llamada Vista Temporal, es la responsable por soportar las especificaciones de las relaciones temporales entre los componentes de un hiperdocumento, definidas estás en una posición relativa del tiempo, como lo muestra en la figura la parte izquierda. En esta vista, los nodos son representados por rectángulos y quienes su longitud indica la duración óptima de su representación en una axisa de tiempo y quienes establecen la relativa posición en un tiempo sincrono.

- La tercera vista, llamada Vista Espacial, soportando la definición de las relaciones de espacio entre los componentes de un documento, estableciendo estas presentaciones como características de un dispositivo determinado, en un instante de tiempo específico. Como lo muestra en la figura (1) en la parte derecha. En esta vista, los rectángulos representan las características espaciales de los objetos, especificando, por ejemplo, su posición en un monitor de video o ene el volumen de un dispositivo de audio en un tiempo determinado.

Las tres vistas están relacionadas con cada una de las mismas, como lo muestra la figura 17. El objetivo en la Vista Estructurada es la base para las cadenas de tiempo en la Vista Temporal. En la vista posterior, para cada punto en el tiempo, la Vista Espacial muestra como los componentes serían presentados en la definición de espacio por los dispositivos de salida. Las ligas y los nodos definidos en la Vista Temporal son inmediatamente actualizados en las Vistas de Estructura y viceversa.

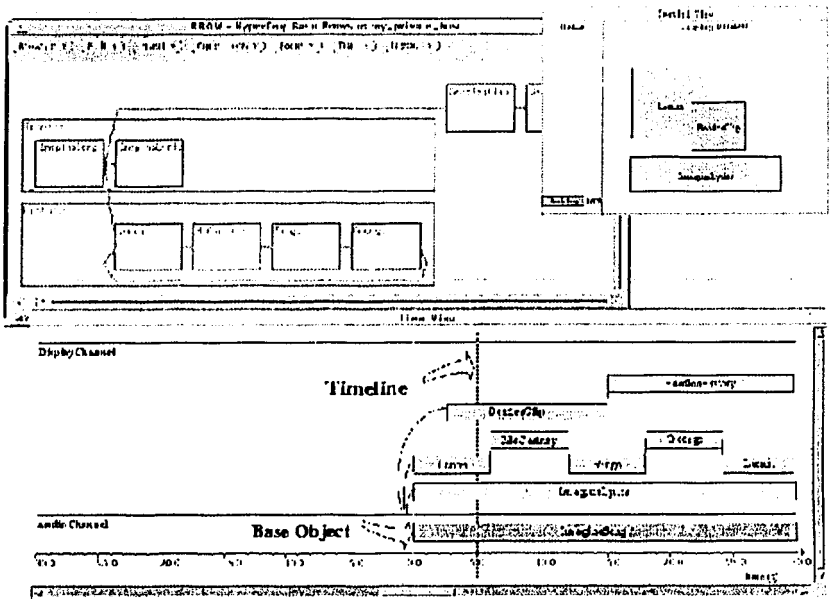


Figura 17 Editor Grafico de HyperProp

El sistema HyperProp en un formato espacial y temporal es responsabilidad por el controlador de la exhibición del documento basado en presentaciones específicas y en la descripción de la plataforma (ambiente). La idea principal es la construcción de un plan o esquema ejecutable para la guía del formato final. La ejecución del plan podría contener información acerca de las acciones que deberían ser lanzadas cuando un evento ocurrente es señalizado.

Nosotros decimos que un evento es predecible cuando este es posible de conocer, a priori, este inicia y finaliza en un tiempo relativo a otro evento, es decir, un evento es llamado no predecible. La secuencia de eventos ocurrentes en el tiempo es llamada cadena de tiempo. Cuando el primer evento de una cadena de tiempo es no predecible, la cadena de tiempo es llamada cadena de tiempo parcial. La ejecución del plan es el conjunto de todas las cadenas de tiempo parciales de un documento. Ésta guía al formato para ajustarse al documento, colocando los componentes del documento en orden para mejorar la calidad de la presentación y reducir las probabilidades de inconsistencia temporal y espacial.

La arquitectura del formato del sistema HyperProp es compuesto por cuatro elementos: el precompilador, el compilar, el ejecutor y los controles de vista (controladores), como se muestra en la figura 18.

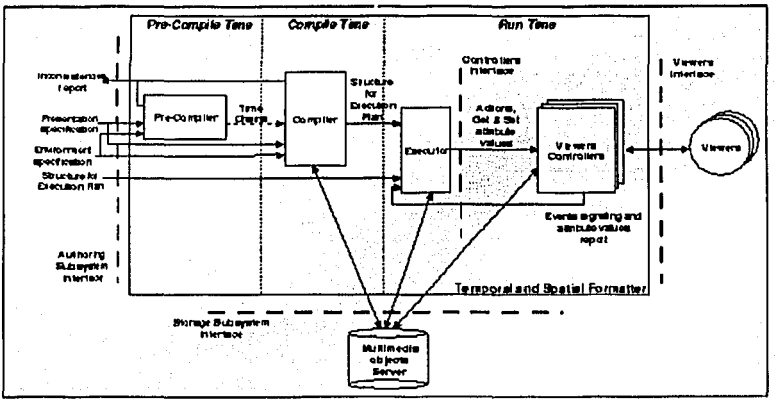


Figura 18 Arquitectura de formato del Sistema HyperProp

Aunque la precompilación se realiza formateando las tareas, éste es en realidad un apoyo al ambiente de autoridad. Su principal función es verificar la consistencia temporal y espacial de cada documento con cadenas de tiempo parciales. Esto ayuda al autor para encontrar los errores, dando al instante una retroalimentación cuando alguna inconsistencia es detectada., como las relaciones de tiempo (inconsistencia temporal), o conflictos con el uso de un dispositivo (inconsistencia espacial) o alguna ausencia de un cierto dispositivo necesitado en la presentación (inconsistencia del ambiente).
Considerando la presentación de un documento completo y la exhibición de una plataforma, el compilador es responsable de la generación de la estructura de datos que se usará por el constructor de formatos ejecutado en el plan. La duración de la presentación del evento en el Modelo de Contexto Anidado es especificado por el descriptor. Uno podría especificar una dupla < t min, t opt, texp, tmax, f cost>, donde el mínimo permitido de duración (t min), el máximo permitido de duración (t max), la duración que sería la mejor en la calidad de la presentación (t opt) y el costo de encogimiento o alargamiento de la duración de un evento (t cost) son definidos. El valor esperado (te exp) es el conjunto inicial para ser igualado al valor optimo. El tiempo de compilación, la duración esperada podría ser computada con base en los costos minimizados en el orden que garantice la consistencia espacial y temporal de un documento.

La implementación en HyperProp de servidores del Modelo de Contexto Anidado (MCN) es responsable por la persistencia del almacenaje de las hyperbases públicas y sus correspondientes nodos descriptores, del contexto de versiones y de rastros públicos. La implementación en general, cuando un usuario pregunta para presentar un documento, el

compilador consigue, desde el servidor NCM, todas los nodos compuestos de nodos recursivos contenidos en la composición que modelan el documento entero. Desde los nodos compuestos, el compilador construye una lista conteniendo todas las ligas que pueden ser cruzadas durante la presentación. Desde estas ligas, las expresiones contenidas en su reunión de puntos son compilados en orden para la obtención concerniente de los eventos y con ellos, crear las cadenas de tiempo parcial. Para cada evento en la cadena de tiempo, el compilador crea una referencia para todas las reuniones de puntos donde estos son usados. Por medio de eso, cuando un evento ocurre el ejecutor conoce todas las ligas que podrían ser evaluadas. Con toda la cadena de tiempo parcial junta, se hace un máquina de estados de la presentación que es alimentada por el ejecutor.

La representación de los objetos son creados por el ejecutor desde su correspondiente descriptor y objetos dato. El ejecutor, sin embargo, no crea la representación del objeto directamente. En su lugar, éste inicia un controlador y pasa toda la información necesaria para crear la representación del objeto y el control de esta presentación entera. La información incluye la descripción del objeto y una lista de todos los eventos que podrían ser reportados, obtenidas de la estructura de datos recibida desde el compilador. El ejecutor conoce el tipo del controlador para ser inicializado, basado en la información contenida en la descripción.

Los controladores crean la representación del objeto obtenida, por la información que manda el servidor NCM. Mediante la asociación del descriptor, los controladores obtienen la lista de operaciones que determinan el comportamiento de los cambios que podrían pasar durante la presentación del objeto. El concepto de un controlador por cada representación de objetos, con una buena interfase definida por el ejecutor, permitiendo incorporarse en exhibiciones comerciales por el sistema, tal como Word, Netscape, etc... Esto únicamente deberá ser implementado por los controladores y sus vistas, permitiendo la transmisión de mensajes acordados por la interfase establecida. Desde las vistas, los controladores reciben la señalización de los eventos reportados por el ejecutor. El ejecutor entonces actualiza apropiadamente la máquina de estados y evalúa todas las ligas asociadas con el evento. Si hay cualquier condición en las ligas que tienen los puntos que satisfagan, las acciones correspondientes se dispararan para su ejecución. En este momento el controlador reporta un evento al ejecutor, este revisa los tiempos esperados en la ejecución del plan y observa si es necesario, se puede ajustar. El ajuste podría ser hecho a través de enviar mensajes, por ejemplo, la aceleración de la presentación.

En la implementación general, el servidor NCM es un procesador centralizado que se comunica con múltiples clientes (sesiones de usuarios) en diferentes máquinas. El ambiente de autoridad, el formato y las vistas son procesos en una simple máquina, que componen una sesión de un usuario. Toda la comunicación (entre procesos de una sesión del usuario y entre las sesiones de los usuarios y el servidor NCM) es implementada sobre el protocolo TCP/IP (en librerías sockets). Este permite, con un pequeño esfuerzo en la implementación una sesión de un usuario distribuido, puede tener controlador / vistas que corren en su máquina, diferente a la maquina donde corre el ejecutor.

C A P Í T U L O 4

EDICIÓN COLABORATIVA Y TÉCNICA PARA EL ADMINISTRADOR DE VERSIONES

4.1 Antecedente de la edición colaborativa

La escritura es una forma de comunicación que se ha utilizado desde que el hombre comienza a razonar sobre cómo comunicarse con otras personas dejando un testimonial de lo acordado. Con el paso del tiempo se empezó a pensar en la forma en que se podría hacer un trabajo en equipo sin importar que las personas se encontraran o no en las mismas instalaciones. Para que los autores escriban conjuntamente, necesitan tener una comprensión compartida del material y así intentar reflejarlo, deben de comprender que tendrán que compartir su texto a los demás autores. Para colaborar los autores dan fuerza a su escritura para enfrentar la naturaleza comunicativa de escribir. La introducción de una herramienta colaboradora de escritura influye en el contexto social de la tarea de escritura, especialmente desde el punto de vista de la negociación para el control, la atención, y significando del texto. Estos aspectos proveen de un espacio de trabajo compartido sobre las computadoras mediante las cuales los autores podrán componer y comunicarse sobre los avances del texto.

La escritura es un pensamiento individual de escritores que trabajan aisladamente con el resto del mundo, la mayoría cuando escriben involucran la interacción social. La colaboración puede tomar forma de correspondencia escrita, que puede ser explícita, pero también incluye todos los textos que están expresados en un universo de discursos escritos que proviene dentro de la herencia de una comunidad. La escritura del individuo puede surgir también fuera de una conversación, en que la palabra escrita llega a ser una extensión directa de un diálogo en proceso con colaboradores. Este es frecuentemente el caso donde hay un autor, pero también hay un número de gente involucrada en discutir un tema de interés común. El texto puede surgir desde la discusión, y fomentar discusión adicional, en un proceso reiterativo. Finalmente, la escritura puede ser explícitamente colaboradora, donde un texto se compone mediante el trabajo conjunto.

Las investigaciones se han encausado al cómo la gente individualmente escribe y qué escribe en grupos. Desde esta investigación un número de teorías de la escritura colaboradora se ha desarrollada. Esto ha inspirado a la vez el desarrollo de las herramientas de computadora para apoyar las diversas tareas que se involucran en el proceso colaborador de escritura.

Los antecedentes teóricos sugieren que la comunicación es un aspecto importante del proceso de escribir conjuntamente. Este aspecto de grupo de escritura se ha incorporado en un número de herramientas a grupos de apoyo que escriben juntos en diferente tiempo, en diferente lugar y a veces ambos un mismo tema.

Como en el campo del CSCW utiliza varias ramas para poder trabajar eficientemente, la edición colaborativa se apoya de la psicología para su funcionamiento, como puede ser el manejo de las dinámicas de grupos que son técnicas para el desarrollo de grupos en un trabajo.

El desarrollo acelerado de las técnicas y teorías metodológicas en el ámbito de la capacitación, plantea una reorientación en la percepción convencional de tales procesos. Hoy ya se requiere de una visión de conjunto que permita ver la capacitación y el adiestramiento no como una evolución de habilidades y conocimientos aislados y esporádicos, sino como una permanente formación integral de los recursos humanos.

Las metodologías que se aplican en las dinámicas de grupos para la coordinación, integración, etc., se han estado trabajando desde la Segunda Guerra Mundial. Ha sufrido ajustes, reforzamientos y variaciones, pero éstos no son muchos en comparación con los que han sufrido sus implementaciones técnicas. Las tendencias de las aplicaciones actualmente son muchas y muy variadas que atacan todos los diversos problemas específicos y organizativos.

4.1.1. Dinámicas de grupo en la Edición Colaborativa

Las dinámicas de grupo como disciplina, estudia las fuerzas que afectan la conducta de los grupos, comenzando por analizar la situación grupal como un todo con forma propia. Del conocimiento y la comprensión de ese todo y de su escritura, surge el conocimiento y la comprensión de cada uno de los aspectos particulares de la vida de un grupo y de sus componentes.

Dentro de las dinámicas de grupos se puede realizar la comunicación, la coordinación, y la negociación que pueden ser utilizadas para facilitar tanto la adquisición de conocimientos como también los cambios de comportamientos y objetivos. Se pone de manifiesto las ventajas del trabajo en grupo sobre el trabajo individual en la toma de decisiones y la solución de problemas.

A consecuencia del manejo de las dinámicas de grupos los textos han aumentado de tamaño, por el manejo de diferentes ideas de los autores y además la complejidad del documento. El documento pasa a ser un objeto que puede tener texto, gráficos y videos para la mejor comprensión con una relación entre ellos, llamada estructura lógica del documento como también lo que nosotros comúnmente utilizamos que es índice, capítulos, figuras, etc.

Se trata de un proceso poco estructurado, difícil de soportar a través de herramientas computacionales rígidas en cuanto a intentar estructurar el proceso.

Las dinámicas de grupo son utilizadas en los ambientes colaborativos por su importancia en lo referente a la conciencia de grupo, en el sentido del manejo de la información por un grupo para llegar a un fin común. En el cual al integrante del grupo se le informa sobre su importancia que tiene dentro del grupo, en sentido de que puede ocasionar retrasos y por consiguiente nunca cumplirse el objetivo planteado.

Para llegar a un modelo de escritura colaborativa Flower y Hays [Flower&Hays,81] identifican tres fases:

- La principal fase dentro del modelo planteado por Flower y Haves es la planificación del documento debido a que en esta etapa se genera toda la información requerida para poder hacer el documento.
    - Dentro de esta fase se encuentra la organización de la información para que no haya ambigüedad y no se repita la información.
    - También se encuentra la declaración de los objetivos o metas que se quieran lograr.
    - Por último con la información, debidamente organizada y planeada, se llega a la división del trabajo para que cada autor desarrolle su parte.
- La segunda fase es la construcción de borradores que son bosquejos de la información que se tiene.
- La última fase que definen los investigadores es la revisión de las partes hechas por los autores, el ensamblaje de todas las partes y la edición del documento.

Como en el campo del CSCW utilizaba varias ramas para poder trabajar eficientemente, dentro de esto en la edición colaborativa se apoya de la psicología para su funcionamiento como puede ser el manejo de las dinámicas de los grupos que es el desarrollo de grupos que en la actualidad es excepcional. Sea cual sea el marco teórico de referencia de la dinámica.

4.1.2 Sistema de edición colaborativa

La edición colaborativa es un proceso en el cual varios autores (editores, revisores, expertos gráficos) con experiencias diversas, interactúan durante la invención y revisión de un documento común. Es una actividad involucrada en la producción de un documento en la que intervienen varios autores.

No se trata de pegar documentos individuales sino más bien de administrar las versiones de un documento en el cual cada miembro del equipo tiene el mismo control sobre el texto y en la interacción. La escritura es una tarea compleja, y recursiva, en la cual se tienen muchas alternativas de diseño, se trata de un proceso poco estructurado, difícil de soportar a través de herramientas computacionales rígidas en cuanto a intentar estructurar el proceso.

Una aplicación colaborativa se puede definir como un sistema computacional que asiste a un grupo de personas dedicadas a una tarea común, cada una de ellas trabajando en su propia computadora, pero compartiendo datos y programas a través de una interfaz multiusuario.

El principal objetivo de las aplicaciones colaborativas consiste en proveer a los usuarios de herramientas que les permitan coordinar sus actividades de trabajo. Así como hay diversidad de definiciones de los sistemas colaborativos, también existen muchos dominios de aplicaciones colaborativas como:

- Aplicaciones que facilitan la interacción a distancia entre miembros de un equipo de trabajo.
- Aplicaciones apoyan la comunicación de personas reunidas físicamente.
- Aplicaciones ayudan a distintas personas en la confección de un mismo documento electrónico.

A pesar de los distintos tipos de aplicaciones que hay, todas tienen un factor común que consiste en hacer más fácil el proceso de compartir información entre un grupo de personas que realizan una tarea común.

La construcción de una aplicación colaborativa es una tarea compleja ya que involucra diversas áreas de trabajo tales como sistemas distribuidos, comunicaciones, interfaces humano-computador, inteligencia artificial, bases de datos y otras áreas como la sociología. La Sociología por ejemplo estudia diversos aspectos del comportamiento humano, principalmente en lo referido a interacciones grupales donde se debe tomar en cuenta distintas características de los integrantes del grupo como la cohesión, las formas de comunicación y retroalimentación, el grado de satisfacción con el trabajo realizado, formas de lograr acuerdos, y otros factores que influyen en la eficacia del trabajo en grupo.

La coordinación de actividades es un aspecto sumamente importante pues determina cómo puede compartirse la información. Es la mejor manera de asignar los recursos y otros aspectos relacionados a objetos comunes. La interacción de un grupo de trabajo puede ser de distintas maneras, lo cual define un estilo de colaboración. En este contexto las reuniones desempeñan un papel muy importante pues permiten la comunicación entre las personas. En las reuniones se analizan muchas veces problemas para los cuales no existe un procedimiento de solución, para ayudar a resolver estos problemas muchas veces se implementan mecanismos de votación, de generación de ideas, de identificación de alternativas, etc. Los Sistemas Colaborativos tienen como finalidad en este sentido aumentar la productividad de toma de decisiones en las reuniones y mejorar la calidad de sus resultados, a través de la estructuración y captura de razonamientos y argumentaciones generadas por los integrantes durante una reunión.

Muchas de las características de la sociedad son adquiridas de las formas en que la gente interactúa. Aunque la presencia de las computadoras en las casas y oficinas es cada día más común, la interacción entre las personas no dista mucho respecto a como lo hacía una década atrás.

Los sistemas computacionales diseñados para apoyar el trabajo individual realizan algunas suposiciones tales como que nadie interfiere, que nadie apura ni observa, ignoran aportes de terceros y no estimulan la formación de consensos. Por el contrario, un sistema colaborativo puede ser visto como un sistema basado en computadora que ayuda en su

trabajo a un grupo de personas que realizan una tarea común y que provee de un ambiente común, permitiendo las aportaciones de terceros.

Dentro de un esquema común de trabajo en grupo, generalmente se establecen roles entre los miembros, ya sean implícitos o explícitos, para hacer más eficiente y coordinado el logro de los objetivos. Por otra parte, la información resultante tanto del proceso de trabajo como del producto final, se conoce como memoria grupal. Además se deben considerar otros factores dentro del esquema de trabajo del grupo, tales como los protocolos de colaboración y la forma en cómo cada uno de los miembros del grupo aprecian el trabajo de los demás, lo que constituye la percepción.

Roles

Un rol es un conjunto de privilegios y responsabilidades atribuidas a una persona o a veces a un módulo de sistema, en este último caso el rol realizado por un software computacional se denomina agente. Los roles podrían ser atribuidos formal o informalmente. Por ejemplo, a una persona a quien le gusta hablar y relacionarse con muchas personas podría informalmente tomar el rol de guardián de la información. El jefe de un grupo podría oficialmente tomar el rol de director del grupo.

Otras complicaciones para el diseño de groupware es que dentro de un grupo, los individuos pueden tomar diferentes roles. De este modo considerando las diferentes parejas que podrían trabajar juntos, tales como: autor/editor, emisor/receptor, orador/oyente y supervisor/subordinado. Note que algunas personas podrían cambiar de rol, ser autor en este momento y después tal vez contribuir en la colaboración como editor. Un software podría soportar no solamente a varias personas trabajando en una tarea, sino también diferentes roles.

Los roles usualmente son utilizados en pequeña o en ninguna medida en el diseño de aplicaciones simples de usuarios como procesadores de texto u hojas de calculo, pero son fundamentales en el diseño de sistemas colaborativos.

Protocolos de Colaboración

Son las distintas maneras de interactuar de las personas en un grupo. Son reglas que permiten a los individuos comunicarse entre sí de tal forma que cada uno pueda enviar y recibir señales comprensibles para los demás, un protocolo de comunicación debe lograr la atención del grupo en el aspecto de la comunicación, identificar los distintos componentes de la comunicación entre las personas, proporcionar retroalimentación constante al grupo de que la comunicación se efectúa satisfactoria o insatisfactoriamente, proporcionar una forma aceptable de concluir la comunicación entre las personas.

Los protocolos grupales o de colaboración pueden ser intuitivos o informales, como los sociales y tecnológicos o formales como control de piso, tomar turnos, etc. Estos tiene su origen en el entorno social y se hacen necesarios siempre que las personas colaboran en una tarea y se transforman en protocolos de acceso a la información cuando éstas lo hacen

mediante el computador, pues el lenguaje, las palabras los gestos se traducen en datos los cuales necesitan ser almacenados, accesados , comprendidos y trasmitidos.

Percepción

Ya que la distancia física disminuye dramáticamente la comunicación entre los miembros de un grupo, se hace necesario crear mecanismos para proveer información sobre la actividad del grupo. Es por esta razón que casi siempre se asocia el concepto de percepción con los sistemas colaborativos en los que no se da la interacción cara a cara, sin embargo los sistemas cara a cara también pueden proveer percepción.

Se entiende por percepción, toda información que provee una conciencia grupal al individuo que forma parte de un grupo. La manera cómo ésta conciencia se obtiene es suministrando información, la cual el usuario interpreta dependiendo de sus requerimientos. Como se refleja intuitivamente, percepción es información sobre la información, se podría hablar de dos tendencias de clasificación, una atendiendo al tipo de la información y la otra atendiendo a la manera en que está información se captura.

En el contexto de la información tenemos percepción de usuarios y percepción de datos. La percepción de usuarios provee información sobre los miembros del grupo, por ejemplo informar quiénes están conectados y lo que estos hacen. La percepción de datos suministra información referente a los cambios efectuados sobre los datos. Parece interesante notar que tanto en el caso a cara a cara como en el distribuido asincrónico, la percepción que casi siempre es requerida es la de datos. La información proveída por la percepción correspondiente al conocimiento compartido por el grupo. Esta se puede proveer por medio de video, texto o despliegue gráfico.

Atendiendo el contexto de la forma de cómo la información se obtiene, la percepción puede ser implícita o explícita. Se puede recoger percepción implícitamente mediante cámaras de video, altavoces, chapas activas, sensores infrarrojos, teclado, mouse o cualquier método que no requiera la introducción explícita de información en torno a la información por parte del usuario. La percepción explícita es proveída por los usuarios en forma de calendarios, horarios, letreros o similares. La decisión de tomar y proveer un determinado tipo de percepción depende de las situaciones que apoyan el sistema colaborativo que se desee implementar y de la estructura de la organización.

Se han realizado diferentes modelos para poder entender los procesos de colaboración, el siguiente modelo nos permitirá describir cualquier sistema de colaboración.

4.2 Modelo de colaboración. AMENITIES [Gea&Gutierrez,01]

En todo proceso que implique colaboración entre personas, deberemos tener en cuenta la estructura y organización del grupo de trabajo. Una teoría que nos permite analizar este proceso es la Teoría de la Actividad, en la cual, una actividad es la unidad mínima con significado en el resultado de la acción de una persona. Estas actividades se realizan para conseguir un objetivo usando una serie de herramientas, y todo ello, dentro de una

comunidad que establece una serie de normas para regular su comportamiento (reglas) y división del trabajo.

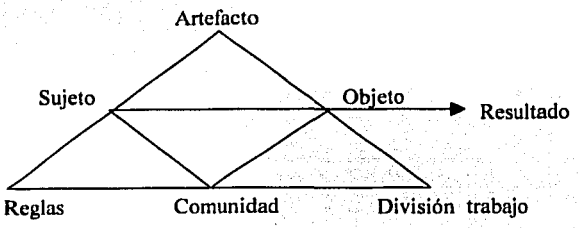


Figura 19 Teoría de la actividad

La figura 19 nos describe como se tienen que dividir los aspectos que interactúan con el ser humano para realizar alguna actividad.

Resultado de todo ello es la importancia de recoger los aspectos organizativos, cognitivos y de interacción que subyace bajo este modelo para adecuarlo a las demandas de cada plataforma en concreto. Por ejemplo, la vista cognitiva nos permite analizar el conocimiento que posee cada participante del entorno colaborativo, y que influye en el proceso de aprendizaje, capacidades para realizar actividades y en la propia conciencia de grupo.

Para abordar la complejidad inherente a estos entornos, se nos da una propuesta de una nueva metodología AMENITIES, que nos permite describir un sistema colaborativo mediante cuatro vistas que facilitan detectar los aspectos más relevantes de este tipo de sistemas que son:

- Vista de grupo
- Vista cognitiva
- Vista de interacción
- Vista de información

Vista de grupo

Primeramente, deberemos identificar los aspectos relacionados con la propia organización (el grupo), y las restricciones que impone esta asociación. Las organizaciones se articulan bajo el concepto de *rol*, que establece la relación entre los miembros del grupo y las tareas que deben llevar a cabo. A menudo esta relación está condicionada por una serie de restricciones impuestas al sistema colaborativo, y de las cuales podemos identificar las siguientes como más importantes:

• Capacidades. Esta es una restricción cognitiva que se impone a cada actor para participar con un rol determinado. Estas capacidades determinan los conocimientos que debe adquirir un usuario para participar con un rol concreto.

• Leyes. Este tipo de restricción viene impuesta por la propia organización e identifica las reglas que deben ser preservadas en el grupo. Normalmente estas reglas se deducen de la propia estructura social que se manifiesta en el grupo (democrática, jerárquica, etc.) Ambas restricciones permiten *modelar sistemas dinámicos*, ya que es habitual que tanto la estructura del grupo como su funcionamiento se modifique en el tiempo (los participantes pueden adquirir nuevas capacidades, variar en número de miembros que lo conforman o bien, modificar las leyes que rigen el grupo al aplicar nuevas estrategias de trabajo).

Vista cognitiva

La vista cognitiva representa el conocimiento que posee o adquiere cada miembro del grupo en el escenario colaborativo. Este conocimiento queda reflejado mediante la descripción de las tareas que puede llevar a cabo.

La descripción de las tareas implica un análisis profundo de las actividades que se deben realizar en el grupo, la división del trabajo y determinar las interrelaciones que existen entre ellas.

El análisis de tareas contempla todos estos pasos, si bien lo hemos dividido en tres fases claramente diferenciadas. En una primera fase definimos lo que denominamos *interfaz del rol*, que recoge las características más relevantes de las tareas a desempeñar por un rol junto a las interrelaciones con el resto de participantes (tareas) y entorno (mediante eventos). Los aspectos más relevantes que identificamos en el interfaz del rol son:

- Identificar tareas a desempeñar

- Relación con otras tareas tales como:
  - Si puede ser interrumpida por otra tarea
  - Su naturaleza cooperativa
  - Mecanismo de activación y modos de sincronización

Este tipo de relaciones modelan el comportamiento tanto del usuario como del propio entorno. Estas relaciones se modelan mediante eventos que provocan cambios en el entorno.

Mediante esta aproximación, nos centramos en el rol como eje central para describir el conocimiento del grupo en lugar de las actividades que se llevan a cabo en el sistema. Esta diferencia nos proporciona una percepción cognitiva (centrada en el usuario y el grupo como ente) en lugar de centramos en flujos de trabajo (orientado a procesos y productos). No obstante, podremos deducir la carga de trabajo del sistema a partir del conocimiento de las tareas de usuarios.

En una segunda fase, se describe y se minimiza cada tarea mediante una descomposición jerárquica y donde se completa con información y aspectos recogidos de otras vistas. En esta descripción de tareas usaremos notaciones que nos permite especificar secuencialidad, concurrencia, optatividad, decisiones, etc.

En una tercera fase, se detalla las tareas tanto individuales como cooperativas, y en las cuales, puede aparece información relativa a otras vistas.

Vista de interacción

Otro aspecto que debemos estudiar son los procesos que implican un diálogo entre participantes para analizar sus características, concretamente:

- El modo de diálogo que se producen entre participantes
- Los requisitos que impone ese diálogo sobre los medios a utilizar

Este modo de diálogo lo identificaremos mediante *protocolos*. Los protocolos se pueden analizar por separado dentro de la organización ya que en gran medida son independientes del dominio del problema, y por tanto, se pueden incorporar al análisis de tareas..Por ejemplo, se pueden identificar protocolos democráticos (toma de una decisión por mayoría), consenso (aprobación unánime de una decisión), jerárquica, etc.

Vista de información

Por último, deberemos recoger la información que es compartida en el escenario. Esta información se puede describir de manera implícita en las actividades y acciones o bien, de modo explícito como flujo de información entre actividades. La información que fluye a través del sistema colaborativo serán los documentos (los objetos que son gestionados en el sistema), eventos y recursos.

Con lo anterior, nos da una idea de que es la edición colaborativa y como diferentes áreas contribuyen para su realización. Adelante se propondrá una técnica para la realización de un administrador de versiones o *versioning* que se explico en anteriores capítulos.

4.3 Técnica para la generación de un administrador de versiones para un sistema de edición colaborativa.

En este apartado del capítulo se propondrá un técnica que nos permita generar un administrador de versiones para un sistema de edición colaborativa.

Los sistemas de edición colaborativa manejan algunos aspectos, tales como:

- Consistencia de los documentos.
- La conciencia de grupo.
- Replicación.

Consistencia de los documentos

Hablando de consistencia, es poder realizar diferentes modificaciones al documento y este aun tenga consistencia en su estructura. Además permitir a diferentes usuarios que se encuentran trabajando en maquinas remotas, poder editar un mismo documento al mismo tiempo.

Cada usuario trabaja de forma remota, por lo que todos los cambios que realicen sobre un documento deben ser visibles para todos los que trabajen sobre él, es decir, se debe contemplar el objetivo de WYSIWIS (*What You see It's What I See*, lo que tu ves es lo que yo veo) y la consistencia del documento compartido. Los sistemas de edición colaborativa utilizan los roles para cuando varios clientes acceden a un mismo párrafo, teniendo el permiso de lectura, lo harán concurrentemente sin problemas de consistencia. Si un autor se encuentra escribiendo y los demás leyendo, también lo podrán accesar al mismo tiempo sin problemas. Pero cuando dos o más autores deseen escribir sobre un mismo párrafo, sólo uno de ellos podrá hacerlo.

La conciencia de grupo.

Como ya se ha explicado en el primer capítulo, la conciencia de grupo es que las personas que se encuentran trabajando en equipo para realizar una meta que en nuestro caso es un documento sepan que alguna modificación que hagan a un documento puede afectar el contexto de otro.

El sistema introduce roles o permisos dentro del mismo, los cuales afectaran a la edición de un documento. Los roles que se tienen son:

- Existen dos tipos de administradores que son:
    - Administrador del Documento, el cual podrá cambiar permisos de los demás autores.
- El rol de escritura: Permite que cualquier autor pueda modificar la estructura de un documento.
- El rol de lectura: Sola mente se tiene el permiso de visualizarlo, pero no puede hacer cambios a la estructura del documento.

Para que un autor pueda realizar cambios en un documento, esté solicita al administrador de documento modificación en sus permisos de su rol.

Replicación

La replicación es una técnica para proveer buen desempeño, alta disponibilidad y la tolerancia a fallas en los Sistemas Distribuidos. Consiste en el mantenimiento en línea de

copias de datos y otros recursos. Un sistema que tenga replicación debe considerar algunos factores que son:

- La transparencia: Los clientes no deben percibir que existen múltiples copias físicas de los datos.

- Consistencia de la información: Es poder realizar diferentes modificaciones al documento y este no sufrir diferencias en su contenido.

Los sistemas de edición colaborativa, hacen replicaciones de los documentos. Las replicas se mantienen en el servidor y en cada una de las maquinas clientes.

4.3.1 Arquitectura de un sistema de edición colaborativa

La mayoría de los sistemas que manejan la edición colaborativa basan su arquitectura en el modelo cliente-servidor. La conexión de un cliente puede ser directamente al servidor usando una dirección IP o desde una INTRANET a través de un servidor PROXY. Se utiliza el protocolo TCP/IP.

La arquitectura que cuenta un sistema de edición colaborativa está formada básicamente por dos elementos: servidor y cliente.

- Servidor. Es que tiene toda la información en una base de datos de la información y un sistema de archivos para el manejo de la información.
-
- Cliente: El cliente manda peticiones al servidor para poder trabajar con un documento.

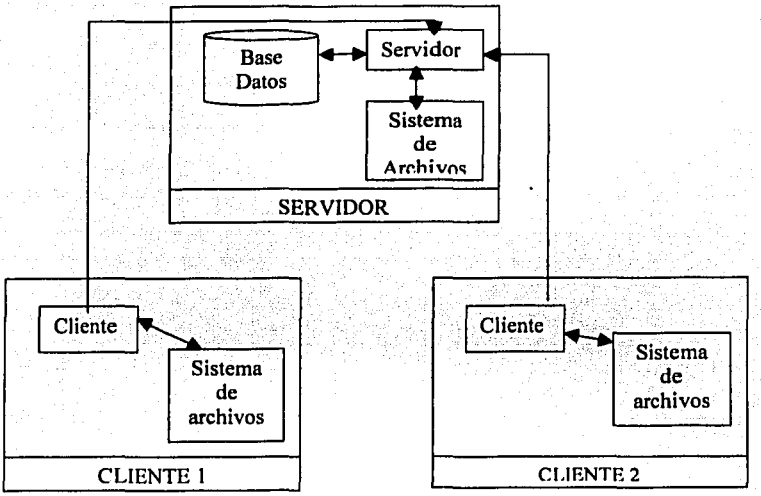


Diagrama 1 Arquitectura de un sistema de edición colaborativa.

El diagrama 1 describe como esta integrado en su arquitectura un editor colaborativo, con dos clientes.

4.4 TÉCNICA PARA LA GENERACIÓN DEL ADMINISTRADOR DE VERSIONES

Para empezar el estudio de la técnica, retomaremos algunos aspectos del modelo Palimpsest, debido a que este se encarga del control de versiones utilizando procesos llamados Deltas. El modelo Palimpsest fue desarrollado por el Doctor David G. Durand para describir el manejo individual de los cambios de los documentos colaborativos.

La técnica que planteamos será una solución a un problema que puede presentar los sistemas de edición colaborativa, debido a que no cuentan con un administrador de versiones para verificar el histórico de las versiones, a que al momento de hacer una revisión y una modificación solamente nos quedamos con la ultima revisión que se ha realizado, eliminando las versiones pasadas o mejor dicho sobrescribiendo la versión actual.

4.4.1 Planteamiento del problema que se tiene en un sistema de edición colaborativa.

Los problemas que se puede encontrar al analizar un sistema de edición colaborativa son:

- El primer problema que se puede encontrar en un sistema de edición colaborativa, es que genere una versión de un documento, sin que está tenga modificaciones en su estructura, tomando en cuenta que cuando se genera una versión es debido a un cambio que se realizo en la estructura del mismo.

- El segundo problema que nos podemos encontrar, que un sistema no cuente con un administrador de versiones para verificar el histórico del documento y tampoco poder observar los cambios que se han realizado sobre el documento.

4.4.2 Propuesta de soluciones para el control de las versiones

La técnica que se propone fue realizada después de un análisis y estudio de diferentes sistemas.

En el aspecto de la generación de las versiones, está debe de tener un límite de generaciones, para que no crezca demasiada y nuestro lugar que tenemos para guardar el histórico se sature, se propone que se puede generar en un documento hasta 1000 versiones. Recordando que se esta trabajando en un ambiente colaborativo y un cambio que se realice dentro de la estructura del documento, tan mínimo sea el cambio (coma, espacio, un acento, etc.) genera una nueva versión del documento.

El esquema general de un sistema de edición colaborativa, es el siguiente:

Paso 1.

El autor debe de ingresar su contraseña y ser aprobado por el sistema desde el servidor, se entra al editor donde podemos apreciar que los documentos, los cuales están cada uno con un identificar dentro de la base de datos.

Paso 2.

Dentro de los documentos, para que nosotros podamos modificar el contenido de los mismos, deben de estar bloqueados, lo cual nos permitirá copiar, borrar o insertar una cadena de caracteres (Strings) o un carácter.

Paso 3.

Al momento, de que ya se haya terminado el bloqueo o terminado el trabajo del autor, uno puede guardar la información del mismo y así crear una versión nueva del documento, ó también se puede salir de la edición sin tener que guardar nada de la información. En este aspecto, se puede encontrar un problema que cuando se bloquea el documento y no se hacen modificaciones a la estructura se guarde, el sistema no debe de crear una versión nueva del documento. Siendo este, un problema para el administrador de versiones.

4.4.2.1 Propuesta a la solución del control de versiones usando Deltas

Comenzamos con dar la propuesta del primer problema, acerca del control de las versiones dentro de un sistema de edición colaborativa:

Al momento que se vaya a guardar la información del documento bloqueado, este debe de hacer una comparación con la versión anterior para ver si hubo cambios y si los hubo entonces generar una nueva versión del sistema. Para ilustrar lo anterior daremos un ejemplo:

UNO V 0
Es un ejemplo, es el documento uno y dice lo siguiente:

Instituto de Investigación de Matemáticas Aplicadas en Computación. Es donde se esta actualmente trabajando en el campo de los Sistemas Distribuidos y el CSCW.

Se bloquea el documento, y se comienza a hacer cambios y son anotados en la forma de color rojo que muestra que hubo cambios.

UNO V 0

Es un ejemplo, es el documento uno y dice lo siguiente:

Instituto de Investigación de Matemáticas Aplicadas en Computación. Es donde se esta actualmente trabajando en el campo de los Sistemas Distribuidos y la edición Colaborativa.

Al momento de guardar el documento, se analiza y encuentra cambios en la estructura del documento y se genera una nueva versión.

UNO V 1

Es un ejemplo, es el documento uno y dice lo siguiente:

Instituto de Investigación de Matemáticas Aplicadas en Computación. Es donde se esta actualmente trabajando en el campo de los Sistemas Distribuidos y la edición Colaborativa.

Como hubo cambios en el documento, en su estructura se hicieron cambios en algunos caracteres por consiguiente se genera una versión que será UNO V1. El mecanismo que lo hace es el siguiente, cuando se bloquea el sistema sabe que se va modificar un texto y lo permite siempre y cuando el autor tenga los permisos de escritora. Al terminar de modificar el texto y se va a guardar, automáticamente el sistema debe de hacer una comparación de caracteres para ver si hubo cambios o no.

Se utilizara funciones incondicionales para hacer la comparación de los String que seria de la siguiente forma:

```
IF ( Doc.V0 = Doc.V0)
   Doc.string.V0 <> Doc.string.V0;
   Doc.V0 = Doc.V0 + 1
 Else
   Doc.V0 = Doc.V0   // en esta parte guardaría la nueva versión
Fin IF
```

Pasada	Caracteres de la Versión Uno V0	Caracteres de la Versión Uno V1	Resultado
1	Es	Es	Igual
2	Un	Un	Igual
3	Ejemplo	Ejemplo	Igual
4	,	,	Igual
5	Es	Es	Igual
..	..	..	..
..	Y	y	Igual
..	El	La	Diferente
..	CSCW	Edición	Diferente
N		Colaborativa	Diferente

Tabla 5 Comparativo con instrucciones de comparación

La tabla 5 nos permite ver como se realiza la generación de una nueva versión del documento, haciendo una comparación de cadena de caracteres o por cada carácter.

Al momento, de hacer las pasadas del documento V0 y el documento V1 notamos diferencias, estas diferencias tan mínimas que sean como un espacio o un carácter o hasta una cadena de caracteres nos generaría una nueva versión debido al cambio que se hizo a la estructura original.

En la parte del control de las versiones generadas por los usuarios, se necesita tener banderas que nos permitan en buen control de las modificaciones, como las usa el modelo

Palimpsest que son Deltas creadas con el propósito de saber que acción se realizo en el documento. Si fue un copiado, una inserción o un borrado de String.

El modelo Palimpsest nos da definiciones sobre las operaciones que son:

Copiado: Esta operación es causada cuando se copian datos de una locación (documento) a otra locación (documento), el copiado hace referencia a dos locaciones (documentos), una fuente y un destino, y el copiado aparece en las dos locaciones (documento).

```
Deltatype copy is {
        Range x: Sequence of Characters;
        Address y: Sequence of Characters;
        Copy ( x, y );
}
```

Borrado: el borrado como lo define el Palimpsest puede ser borrado o removido. El borrado elimina un dato de la estructura. El remover un dato de una locación (documento) particular, pero no previene que el dato fue una fuente de una operación de un copiado, lo cual causa que el dato este incluido en el destino. Palimpsest analiza este movimiento del remover un dato y lo que propone es que sea insertado en otro documento.

```
Deltatype deletion is {
    Range x: Sequence of Characters;
    Delete ( x );
}
```

Además de las anteriores Deltas se propusieron deltas para diferentes operaciones, con las cuales nos ayuda para el manejo de las estructuras de los documentos.

```
Deltatype text-node is {
    Data contents: Sequence of
Characters;
    Provide (contents);
}

Deltatype move is {
        Range x: Sequence of Characters;
        Address y: Sequence of Characters;
        Copy ( x, y );
        Remove (x );
}

Deltype insert is {
   Data insertion-contents :  Sequence of Characters;
   Address where : Sequence of Characters;
   Copy ( insertion-contents, where );
}
```

```
Deltatype link is {
  Data link_contents = Sequence of {
          Data from : Reference;
          Data to : Reference;
          Data explainer : Sequence of Characters;
    }
}
```

4.4.2.2 Otra propuesta a la solución del control de versiones usando XML.

La nueva propuesta se realiza debido al manejo de la información, que ya no se ocuparían deltas para indicar los procesos que se efectúan para el manejo de la información y que esta nos permite optimizar recursos de los equipos. Seria con el manejo de los DTD[14] del lenguaje XML[15].

Como recordemos que los documentos en algunos sistema de edición colaborativa, están realizados por medio de XML. Lo que nos permite poder manipular el documento como mejor nos convenga.

Dentro del manejo del XML, nosotros podemos definir diferentes tipo de documentos que son los DTD.

Lo que se propone es la creación de DTD para la adicionar comentarios acerca de cuando se ejecute una modificación en la estructura de los documentos.

Para comenzar se tiene que explicar lo que es el XML.

El lenguaje HTML es un subconjunto de XML especializado en presentación de documentos para la Web, mientras que el XML es un subconjunto de SGML[16] especializado en gestión de información para la Web.

XML puede utilizarse para crear documentos de texto que contienen datos con un formato estructurado. Además de los datos, puede incluirse un conjunto detallado de reglas que definen la estructura de dichos datos. El autor de documento XML define esas reglas.

Los DTD que utiliza XML son los que integran cada editor para validar etiquetas existentes, atributos, valores, etc., haciendo posible que se vayan integrando en el código del documento únicamente de acuerdo con dichas normas. Se puede decir que un DTD es

[14] DTD Definidor de Tipos de Documentos
[15] XML Extensible Markup Language, Lenguaje de marcas extensible
[16] SGML Standard Generalizad Markup Language, lenguaje en marcas generalizado estándar

una definición exacta de la gramática de un documento, con la misión de que se genere el código adecuado sin errores.

Dentro del XML no existen DTD predefinidas, por lo que se debe de diseñar sus propios DTD para cada tipo de documento XML.

Los DTD manejan diferentes componentes para su construcción, como son:

<!ELEMENT Nombre del elemento regla>

Siendo !ELEMENT sirve para la declaración de las reglas que se van a utilizar para validar el documento. La regla define la regla para el contenido del elemento.

Dentro de las reglas, el DTD tiene algunos predefinidos que son:

#PCDATA: Se utiliza cuando el contenido de un elemento es texto.

Ejemplo:

< !ELEMENT titulo (#PCDATA) >

ANY: Incluye tanto texto como elementos secundarios en su contenido.

< !ELEMENT HTML ANY >

Con esto permite incluir información dentro de la etiqueta <HTML>...</HTML>

Como XML no cuenta con DTD predefinido es una gran ventaja, ya que proporciona una total libertad de adecuación a cada documento.

Además de contar con DTD, el XML maneja el XSL[17] que nos permite especificar como deseamos presentar los datos de un documento XML, sino también sirve para filtrar los datos de acuerdo a ciertas condiciones. El XSL nos permite la descripción de la presentación física del documento XML, si no, también posibilita la ejecución de bucles, sentencias IF.. THEN, selecciones por comparación, operaciones lógicas, etc..

Ya teniendo la información acerca de lo que se puede realizar con XML. Se propone la creación de un XSL para cuando sea modificado la estructura del documento.

Como XSL puede hacer un análisis del documento y se pueden manejar condiciones. Por lo tanto, el proceso que se manejaría sería:

Proceso.

1.- Al momento, que se este editando el documento.

[17] XSL Extensible Stylesheet Language, Lenguaje de hojas de estilo extensible.

2.- Como esta en XML, se crearía un nuevo DTD, para el guardado de los cambios que se hicieran sobre el documento.

3.- Lo que haría el DTD, es guardar la información de los cambios realizados a la estructura del documento, como ejemplo.

<!ELEMENT Inserción (#PCDATA) >
<!ELEMENT Borrado(#PCDATA)>

Con la creación del DTD, nos utilizar luego un programa realizado en XSL.

4.- Lo que haría el XSL, sería el manejo de la información de los procesos de los elementos del DTD, que son:

- Inserción
- Borrado

La forma que se manejaría sería:

<xsl:template match="/">

<xsl:if Inserción="Información">

Lo que nos indicaría es que si hay información dentro de PCDATA de Inserción, entonces se realizo una inserción dentro de la estructura del documento.

Lo escriba en el campo Inserción.

Inserción:

Lo que nos indicaría es que si hay información dentro de PCDATA de Borrado, entonces se realizo una inserción dentro de la estructura del documento.

Lo escriba en el campo Borrado

<xsl:if Borrado="">

Borrado:

Se seguiría manejando que un cambio dentro de la estructura del documento generaría nueva versión. Con la última novedad, que cuando se genere la versión, se adicionaría en el documento XML comentarios acerca de los cambios que se hicieron.

Permitiéndonos que cuando queramos visualizar el documento, este ya mostraría la información de la inserción de caracteres, como también que información se borro.

Dándonos la facilidad de uso, para dejar de hacer la comparación de versión en el administrador.

4.4.3 Propuesta a la solución del administrador de versiones

El siguiente problema que nos encontramos es la falta de un administrador de versiones en un sistema de edición colaborativa, que nos servirá para verificar el histórico de las versiones, como también poder compara versiones, y eliminar versiones pasadas.

Dando una explicación de un administrador de versiones, es el poder manejar la información de las versiones que se tengan de un documento. La generación de una versión es debida a un cambio en la estructura de un documento que puede ser tan mínima como sería la inserción de una coma, un punto, un espacio, hasta la inserción de varios párrafos o de el borrado de párrafos. Por tal motivo, se generaría una versión. Para luego teniendo un administrador de versiones poder observar el histórico de las versiones que se han generado de un documento.

No se debe de olvidar que se esta trabajando en una ambiente colaborativo donde varias personas interactúan para llegar a un fin común y que se pueden encontrar en diferentes partes del mundo y que pueden trabajar en un mismo tiempo o en un tiempo desfasado entre ellos.

Enseguida se dará la propuesta para la solución de este problema, siguiendo el mismo contexto de información.

El esquema siguiente es la propuesta para el administrador de versiones:

Paso 1.

Para poder utilizar el administrador de versiones, se deberá estar utilizando un documento.

Paso 2.

Como el administrador de versiones llamado posteriormente AV, será una herramienta integrada al editor, para empezar a trabajar con el AV se deberá de pedir en un cuadro de diálogos un usuario y una contraseña, al momento, de que se firme, este se mandara al servidor para que coteje las firmas y los permisos del autor. Existirán dos tipos de permisos:

- El administrador, será la persona que podrá trabajar con las tres opciones: Histórico, Comparación y Eliminación.
- El autor, únicamente podrá acceder a dos opciones y son: Histórico y el de comparación.

La creación de dos tipos de permisos ayudara a que no cualquier persona pueda eliminar versiones pasadas.

Paso 3.

Si la persona que firma es validado por el Servidor, se desplegara algunas opciones de la herramienta tales como:

- **Histórico**. Si el autor se mete en la opción del histórico, esta deberá desplegar una ventana con el histórico de las versiones que se han efectuado del documento, además de mostrar quien lo modifico, la fecha de cuando se hizo la modificación, y una ordenación del histórico.

- **Comparación**. Si el autor entrar a esta opción, el administrador de versiones podrá ver la versión pasada y la nueva versión. Con un color diferente al original se mostrarán las modificaciones que se han realizado.

- **Eliminación**. (Únicamente el administrador) Permitirá al administrador poder eliminar una versión atrasada o una versión que se creo pero sin modificaciones relevantes para el documento.

Paso 4.

Cuando el usuario escoja alguna de las opciones se realizarán operaciones diferentes. Y estas opciones estarán interactuando directamente con el servidor, para poder ver las ultimas actualización de las versiones de los documentos.

Es necesario la construcción de una tabla en la base de datos que tiene el servidor. Para poder tomar valores de la misma para generar una llave para poder ubicar las versiones de un documento.

La estructura de la tabla versiones es la siguiente:

Campo	Tipo	Longitud
ID _ ARCHIVO	Numérico	2
ID _ DOCUMENTO	Numérico	4
VERSION	Flotante	--
NOMBRE _ USUARIO	Numérico	4
MODIFICADO	Date	8
COMENTARIOS	Carácter	100

Los campos que se proponen se explican enseguida:

ID_ARCHIVO: El campo que se requiere de este, es el identificador del documento, este campo esta relacionado con otra tabla de la base de datos y de la cual obtendremos el identificador completo del documento.

ID_DOCUMENTO: El campo que se requiere de este, es el identificador del documento, este campo esta relacionado con otra tabla de la base de datos y de la cual obtendremos el identificador completo del documento.

VERSION: El campo que se requiere de este, es el identificador de la versión, para que se pueda hacer si es que hay cambios, generar una nueva versión y sea guardada en su respectiva base. Este campo esta relacionado con otra tabla la cual obtendremos la versión en la que se encuentra el documento.

NOMBRE_USUARIO: El campo que se requiere de este, este campo esta relacionado con otra tabla la cual contiene la información de todas las personas que están registrados dentro del sistema y así poder tener un mejor control de acerca de quien fue la persona quien bloqueo el documento.

MODIFICADO: El campo que se requiere de este, es al momento de que se guarde el documento y si modificaron las banderas de los deltas, entonces que guarde la fecha de la nueva versión.

COMENTARIOS: el campo que se requiere de este, es al momento del guardado para que mande la información de que fue lo que realizo el autor en el documento.

La tabla que se cree deberá estar relacionada con algunas otras tablas de una base de datos que estará funcionando en el servidor del sistema de edición colaborativa, las cuales serían:

a) La tabla de los usuarios. Para obtener el registro del nombre_autor.
b) La tabla del documento. Para obtener los registros del id_documento y el de la versión.
c) La tabla archivo. Para obtener el registro id_archivo y el registro de modificado.

Para que nosotros podamos utilizar la información para obtener una versión debe de contener la llave con la cuál nosotros podemos identificar la versión que necesitamos.

Su proceso sería:

Paso 1. Se genera un evento en el administrador de versiones.

Paso 2. Al pedir una versión al servidor

Paso 3. Manda una solicitud que contendrá la siguiente llave:
a. {id_archivo, id_documento, versión, nombre_usuario, modificado}

Paso 4. El servidor lo busca en la base de datos y luego busca en el sistema de archivo la versión con la llave y la manda a la maquina remota.

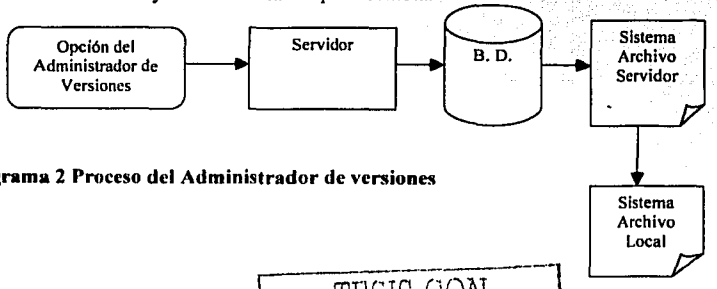


Diagrama 2 Proceso del Administrador de versiones

Enseguida se mostrara el diagrama de flujo del administrador de versiones.

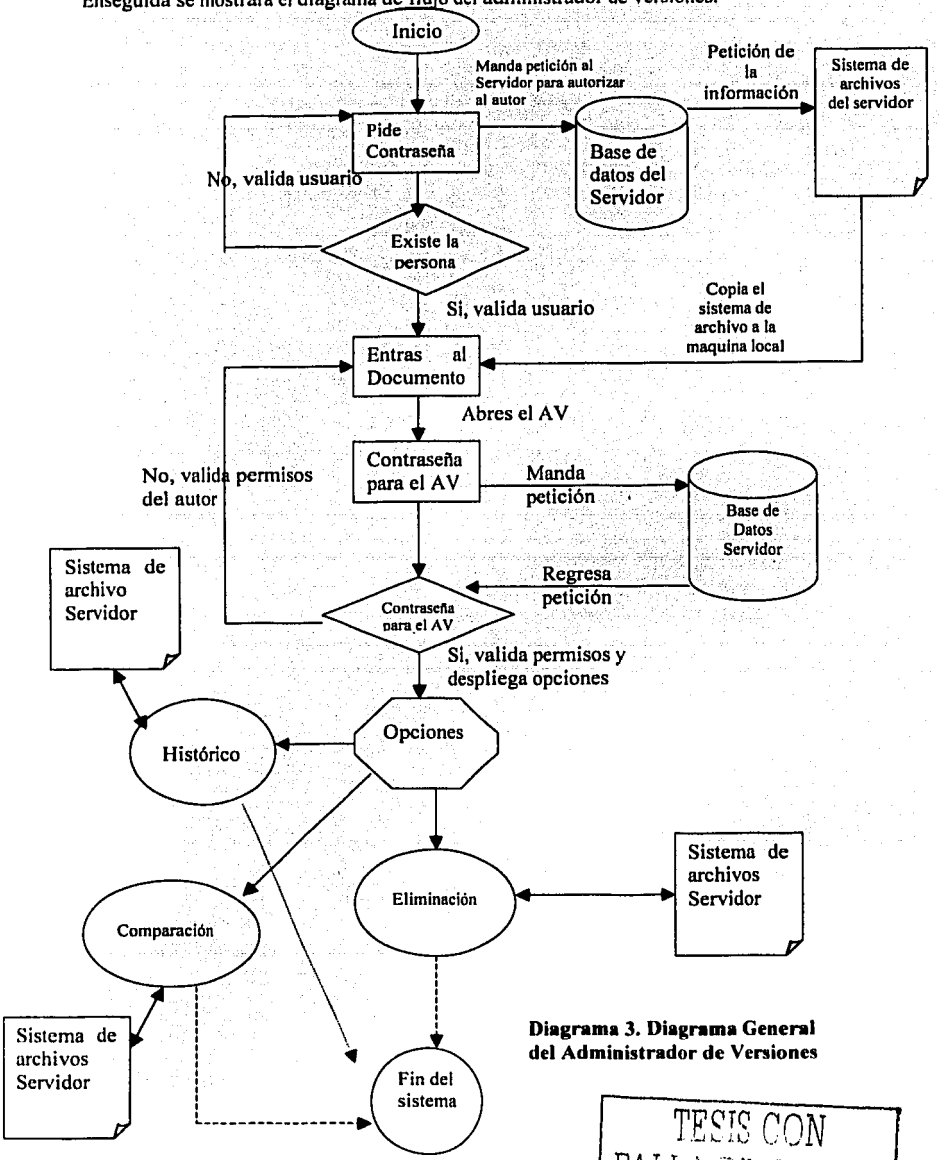


Diagrama 3. Diagrama General del Administrador de Versiones

El diagrama 3, nos indica que al momento de que sea aceptado el usuario y la contraseña, podrá entrar a trabajar con el administrador de versiones. Esta solicitud nos permite tener un mejor control en la revisión de las versiones que se han realizado en un documento.

Tomando en cuenta dos tipos de permisos dentro del administrador de versiones que son:

- El permiso total, que solamente el administrador del documento podrá entrar a la opción de eliminación de una versión.

- El permiso normal que lo tendrá un autor, que únicamente no podrá entra a la opción de borrado de una versión.

4.4.3.1 Histórico

El procedimiento para realizar esta opción es la siguiente:

1- Selecciona el documento para ver sus versiones.

2- Se activa el evento al dar un click en la opción.

3- Se reestablece la comunicación con el servidor.

4- La petición la recoge la Base de Datos y la manda a la tabla versión, para sacar la llave.

   - La llave contiene {id_archivo, id_documento, versión, nombre_usuario, modificado}

5- Al tener completa la llave el servidor, pasa al sistema de archivo del servidor para visualizar la información que se tiene.

Archivo	Documento	Versión	Usuario	Modificado
Tutorial	Uno	1	Manuel	12/12/02
		2	Edgar	10/12/02
		3	Cesar	09/11/02
		4	Silvia	09/11/02
		5	Luis	01/11/02
		6	Rocío	25/10/02
		7	Araceli	15/10/02
		8	Edgar	30/09/02
		9	Manuel	25/09/02
		10	Cesar	23/09/02

Su diagrama de flujo es el siguiente:

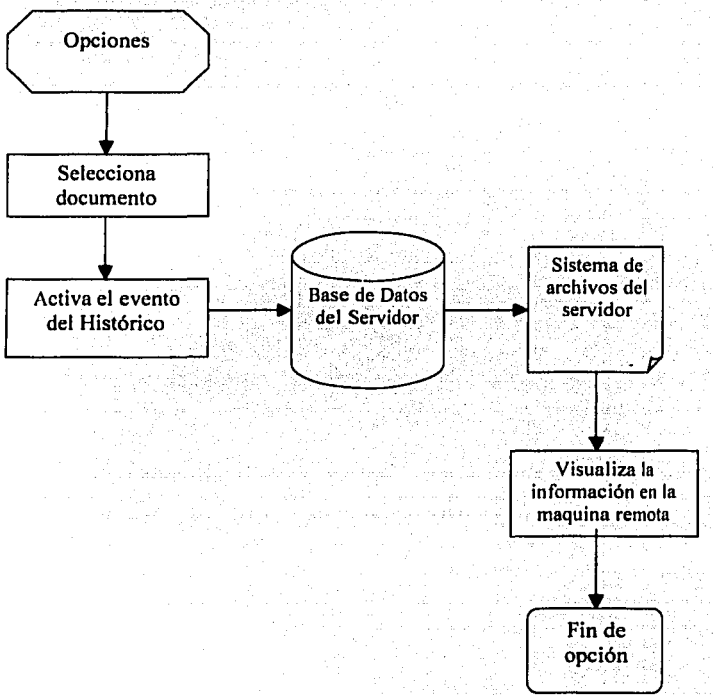


Diagrama 4 Histórico

El diagrama 4 nos muestra el flujo de información que se lleva en el proceso del histórico.

4.4.3.2 Comparación

El procedimiento para realizar esta opción es la siguiente:

1.- Se activa el evento al dar un click en la opción.

2.- Se activa la conexión con el servidor.

3.- Selecciona el documento para ver sus versiones.

4.- La petición la recoge la Base de Datos y la manda a la tabla versión, para sacar la llave.

- La llave contiene {id_archivo, id_documento, versión, nombre_usuario, modificado, comentario}

5.- Al tener completa la llave el servidor, pasa al sistema de archivo del servidor y muestra la información de la llave con respecto a la llave.

Archivo	Documento	Versión	Usuario	Modificado	Comentario
Tutorial	Uno	1	Manuel	12/12/02	Se modifico el párrafo 5, se inserción: el siguiente texto: Edición Colaborativa Borrado: ---
		2	Edgar	10/12/02	
		3	Cesar	09/11/02	
		4	Silvia	09/11/02	
		5	Luis	01/11/02	
		6	Rocío	25/10/02	
		7	Araceli	15/10/02	
		8	Edgar	30/09/02	
		9	Manuel	25/09/02	

6.- En la pantalla aparece una nueva columna que nos indica que acciones se realizarón en la estructura del documento.

Su diagrama de flujo es el siguiente:

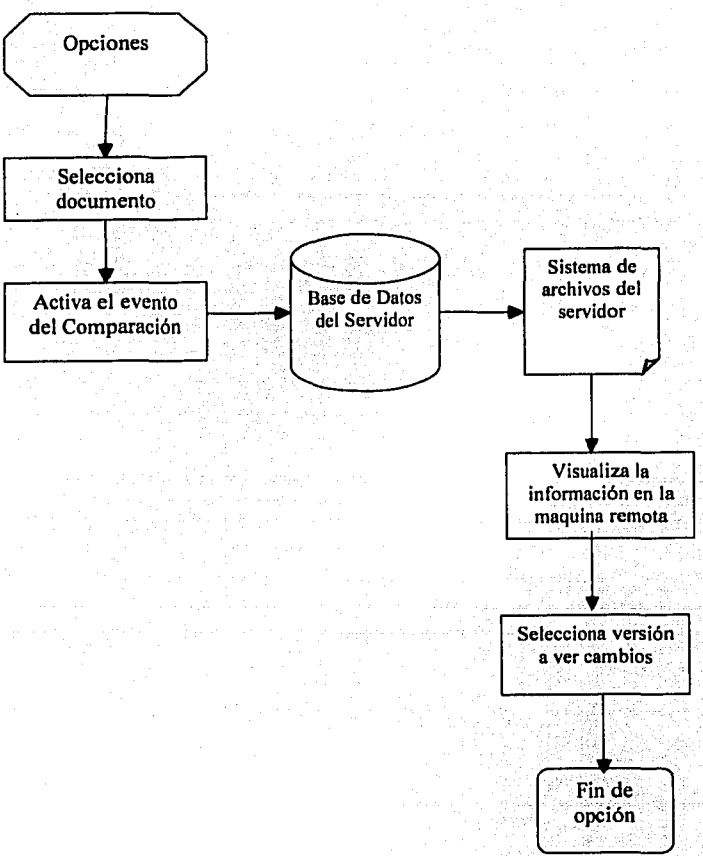


Diagrama 5 Comparación

En diagrama 5 muestra como es el proceso para mostrar la comparación de versiones.

4.4.3.3 Eliminación

El procedimiento para realizar esta opción es la siguiente:

1.- Se activa el evento al dar un click en la opción.

2.- Se activa la conexión con el servidor.

3.- Selecciona el documento para ver sus versiones.

4.- La petición la recoge la Base de Datos y la manda a la tabla versión, para sacar la llave.

- La llave contiene {id_archivo, id_documento, versión, nombre_usuario, modificado}

5.- Al tener completa la llave el servidor, pasa al sistema de archivo del servidor y muestra la información de la llave con respecto a la llave.

Archivo	Documento	Versión	Usuario	Modificado
Tutorial	Uno	1	Manuel	12/12/02
		2	Edgar	10/12/02
		3	Cesar	09/11/02
		4	Silvia	09/11/02
		5	Luis	01/11/02
		6	Rocío	25/10/02
		7	Araceli	15/10/02
		8	Edgar	30/09/02

6.- Para eliminar una versión, solamente seleccionando la versión a borrar.

7.- La petición se manda al servidor, y lo elimina de la base de datos y del sistema de archivo.

Su diagrama de flujo es el siguiente:

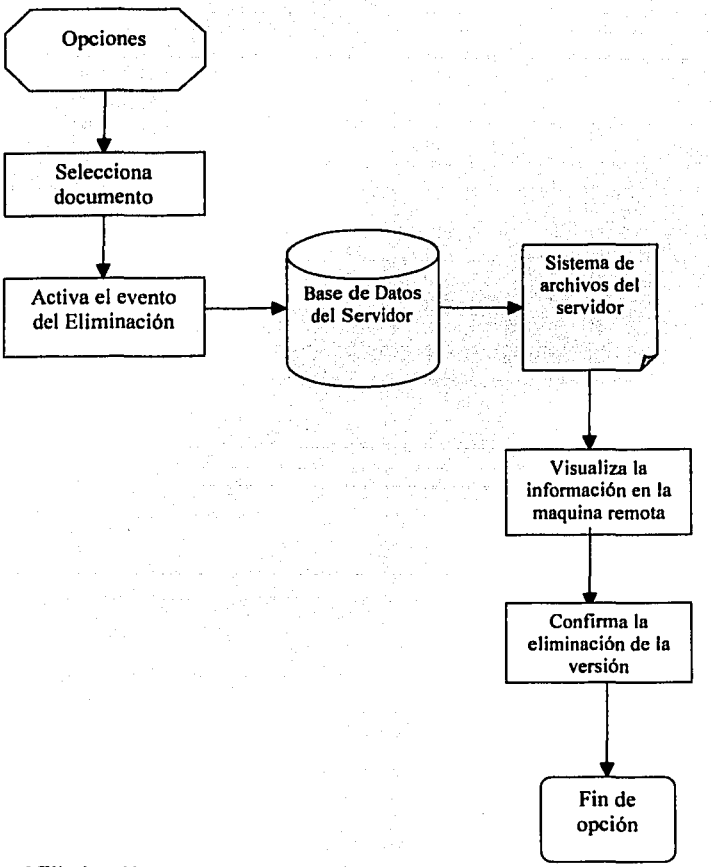


Diagrama 6 Eliminación

El diagrama 6, muestra que se selecciona la versión que se desea borrar, la cual al aceptar su eliminación se borra en el sistema de archivo y su registro en la base de datos del servidor.

4.4.4 Algoritmo del Administrador de Versiones.

1- Inicio

2- Declaran variables
admón., vector1, vector2, vector3
id_Doc, , id_ver
delta_c, delta_b, delta_i
usua, contra

3- Selección de la opción del Administrador de versiones

4- Solicita Usuario y Contraseña, variables Usua y Contra

5- La información del Usua y Contra manda petición a la base de datos para que revise en la tabla de usuarios si tiene los permisos necesarios el autor para entrar al menú.

6- Regresa petición, si es aceptada entra a un proceso de selección.

7- Si el usuario le da en Histórico

8- Se registra en una variable admón.

9- Utiliza una condición While que almacena la petición del evento.

10- Case Histórico

11- Manda una petición a la base de datos para recoger la siguiente llave que contiene {id_archivo, id_documento, versión, nombre_usuario}.

12- Manda petición con la llave al sistema de archivo, para recoger los archivos que tiene la llave

13- Regresa información completa con datos del día de las modificaciones y el autor quien lo realizo.

14- La información es desplegada en pantalla.

Archivo	Documento	Versión	Usuario	Modificado
Tutorial	Uno	1	Manuel	12/12/02
		2	Edgar	10/12/02
		3	Cesar	09/11/02
		4	Silvia	09/11/02
		5	Luis	01/11/02
		6	Rocío	25/10/02
		7	Araceli	15/10/02
		8	Edgar	30/09/02
		9	Manuel	25/09/02
		10	Cesar	23/09/02

15- Para visualizarlo, solamente con doble clic y solamente se visualizara en solo lectura, para que no se realicen cambios.

16- Break; rompe el caso

17- Regresa a espera de un evento de selección del administrador de versiones

18- Si se da en el evento de Comparación

19- Case Comparación

20- Pide a la Base de Datos información que son extraídos de la tabla versión para sacar la llave {id_archivo, id_documento, versión, nombre_usuario, modificado, comentario}

21- Completa la información de la llave, manda la siguiente petición al sistema de archivo, para recoger los archivos que cumplan con la llave.

22- Despliega en pantalla todos los encabezados de las versiones que tenga el documento.

23- Al tener la versión seleccionada se utiliza la información de la columna comentario para observar las modificaciones hechas a la versión.

Archivo	Documento	Versión	Usuario	Modificado	Comentario
Tutorial	Uno	1	Manuel	12/12/02	Se modifico el párrafo 5, se inserción: el siguiente texto: Edición Colaborativa Borrado: ---
		2	Edgar	10/12/02	
		3	Cesar	09/11/02	
		4	Silvia	09/11/02	
		5	Luis	01/11/02	
		6	Rocío	25/10/02	
		7	Araceli	15/10/02	
		8	Edgar	30/09/02	
		9	Manuel	25/09/02	

24- Break, hasta que el autor salga de la opción

Solo se podrá visualizar, sin poder modificar ninguna versión.

25- Si se da en el evento de Eliminación

26- Case Eliminación

27- Manda el contenido de la llave contiene {id_archivo, id_documento, versión, nombre_usuario, modificado} a la base de datos.

28- Del cual pasa la petición al sistema de archivo del servidor.

29- Regresa información con referencia a las versiones del documento.

30- Al seleccionar una versión se genera un evento

31- El evento manda una petición al sistema de archivos y a la base de datos, para eliminar la versión.

32- El evento indicara si se quiere eliminar o no la versión.

33- Si se elimina, manda a la base de datos y al sistema de archivos del servidor, para su borrado. Por tal motivo, es importante que la persona que haga el borrado del archivo sea un administrador o el autor principal del documento.

34- En el sistema de archivos borra el archivo y en la tabla de versiones borra el registro de la versión.

Archivo	Documento	Versión	Usuario	Modificado
Tutorial	Uno	1	Manuel	12/12/02
		2	Edgar	10/12/02
		3	Cesar	09/11/02
		4	Silvia	09/11/02
		5	Luis	01/11/02
		6	Rocío	25/10/02
		7	Araceli	15/10/02
		8	Edgar	30/09/02

35- Break, hasta que el autor salga de la opción

36- Regresa al Editor.

C O N C L U S I O N E S

El trabajo presenta una propuesta de un administrador de versiones para el sistema de edición colaborativa. Establece principios que pueden ser importantes para el desarrollo del administrador de versiones y de los cambios que se realizan a estos. Además de dar una explicación sobre que es la edición colaborativa y como interactúa con diferentes áreas del conocimiento.

Debido al gran desarrollo que a tenido el uso del Internet, no solamente para correos electrónico ahora para el negocio en línea o una videoconferencia, nosotros tomamos el principio de las videoconferencias que se encuentran ubicadas en el campo del CSCW (Trabajo Colaborativo Asistido por Computadoras), donde encontramos el principio de la edición colaborativa. Esto nos lleva a realizar un trabajo para el administrador de versiones que es soportado por medio del Internet.

Hemos visto que un buen administrador de versiones ayudaría bastante a un grupo de personas que se dediquen a la obtención de documentos estructurados como pueden ser artículos científicos, libros, etc. Debido que nos permite trabajar con otras personas en distintos lugares y tiempos, sin afectarles económicamente porque todo lo resolverían por medio del Internet.

Un problema que tuvimos fue hecho que la información recopilada para la elaboración de este trabajo de titulación fue bastante difícil de encontrar debido a que no se encuentran libros sobre el tema de control de versiones. Toda la información como son los modelos y los sistemas fueron sacados de artículos de investigadores por medio de Internet.

La información encontrada fue principalmente artículos de investigadores extranjeros, provocando una traducción de la información y luego un análisis del mismo, para poder plasmar en el trabajo la información obtenida.

Está información es una parte importante del proyecto que puede ser de gran utilidad para otras personas para explorar el campo de la edición colaborativa, dando en que parte se encuentra el tema en los sistemas distribuidos, además de incorporar modelos y sistemas que se han trabajado para la edición colaborativa.

Otro problema que nos enfrentamos fue el hecho de que los sistemas que incorporaban el control de versiones la mayoría se encuentra en forma experimental, solamente encontrando un sistema de libre uso que es el CVS.

La propuesta que se entrega en este trabajo, servirá para una futura investigación para la obtención del administrador de versiones para sistemas de edición colaborativa, así como la información que se recopilo para desarrollar el proyecto.

La experiencia obtenida en este trabajo es muy grande y satisfactoria, debido a que se emplearon algunos principios de las materias que se cursaron en la carrera como fueron teleprocesos, seminario de titulación II, sistemas operativos y parte de las materias de

programación y de matemáticas, para poder desarrollar un análisis del sistema de edición colaborativa. Además de tener que forzarme a conseguir la información por otros medio que no fuera la biblioteca y que no estuviera en mi idioma, teniendo que traducir la información en Inglés y en francés.

La aportación que se da, es una base para el manejo de la edición colaborativa y los principios para un administrador de versiones.

A N E X O

ANEXO DE FIGURAS

ANEXO DE TABLAS

ANEXO DE DIAGRAMAS

B I B L I O G R A F Í A

[Twidale&Nichols,98] A Survey of Applications of CSCW for Digital Libraries
Michael B. Twidale
David M. Nichols; 1998

[Flower&Hays,81] Understanding Systemic Change: Using the Web in Collaborative Research; Collaborative Writing Research
Flower & Hays

[Pfeifer et al,95] Electronic Meeting CSCW – GDSS
John Pfeifer
King Koo
Pricilla Yin 1995

[Tanenbaum,1] Sistemas Operativos Diseño e Implementación
Andrew S. Tanenbaum
Prentice – Hall

[Tanenbaum,2] Sistemas Operativos Distribuidos
Andrew S. Tanenbaum
Prentice -Hall

[Tanenbaum&Van Renesse,85] Sistemas Operativos Distribuidos
Andrew S. Tanenbaum
Van Renesse
Prentice - Hall

[Mullender,89] Distributed Systems
Sape Mullender
Frontier Series ACM Press 1989

[Coulouris&Jean,88] Distributed Systems Concepts and Design
George F. Coulouris
Jean Dollimore
Addison – Wesley Publishers; 1988

[Vitali&Durand,94] Using Versioning to support collaboration on the WWW
Fabio Vitali
David G. Durand 1994

[Ellis et al, 93] Some Issues and Experiences
C. A. Ellis
S. J. Gibbs
G. L. Rein 1993

Publicadas: Groupware: Software for Computer – Supported Cooperative Work
IEEE Computer Society Press
Morgan Kuafmman Publishers, San Francisco California 9-28

[Bock,93] Groupware: The Next Generation for Information Processing
Geoffrey Bock

Publicadas: Groupware: Software for Computer – Supported Cooperative Work
IEEE Computer Society Press
Morgan Kuafmman Publishers, San Francisco California 1-8

[Bush,45] As We May Think
Vannevar Bush
Atlantic Monthly, Mayo 1945

[Coleman,95] Groupware, Technology & Applications
David Coleman, 1995

[Lynch et al,90] Supporting Collaborative Research on International Technology Trends
Lynch, K., Snyder, J. y Vogel, D. 1990,

[Johnason,88] Groupware: Computer Support for Business Teams.
Johanson, R., 1988
The Free Press.

[Capítulo 2] Modelos de Versiones
Edgar Guzmán

[Durand,94] Palimpsest: A data Model for Revision Control
David G. Durand 1994

[Whitehead et al,94] A Proposal for Versioning Support for the Chimera System
E. James Whitehead, Jr.,
Kenneth M. Anderson
Richard N. Taylor 1994

[Soares&Rodriguez,94] Nested Composite Nodes and Version Control in
Hypermedia Systems
Luiz Fernando G. Soares
Noemí L.R. Rodríguez 1994

[Amsden&Clemm,99] Web Versioning Model
Jim Amsden
Geoff Clemm 1999

[Whitehead,98] Collaborative Authoring on the Web: Introducing WebDAV
Jim Whitehead, Jr., 1999
Publicada: Bulletin of the American Society for Information Sciencia Vol. 25 No. 1
Octubre / Noviembre 1998

[Whitehead&Meredith,98] WebDAV: IETF Standard for Collaborative Authoring on the Web.
E. James Whitehead
Meredith Wiggins 1998

[Rada,96] The Many Using & Creating Hipermedia (MUCH) System
Roy Rada, 1996

[Rada&Chen,96] Modelling Situated Actions in Collaborative Hypertext Databases
Roy Rada
Chaomei Chen

Revista: Journal of Computer – Mediated Communication
Volumen 2, Numero 3 Diciembre,1996

[D'Cantu,80] Probabilidad y Estadística para ciencias químico – biológicos
María Jose Marques D'Cantu
McGraw - Hill

[Stewart,94] Position on Versioning in Hypertext System
Linda Stewart,
ECHT'94 Septiembre 1994

[Eaton,94] Advantages Migrating to ClearCase Version 2
David W. Eaton Septiembre 1994

[Peña,98] Documentación de CVS
Javier Fernández – Sanguino Peña
Marzo 1998

[Signum,98] CVS – Concurrent Versions System
Signum Support AB

[Dreilinger,98] CVS Version Control for Web Site Projects
Sean Dreilinger 1998

[Lee,97] CVS – A Project Based Version Control System
Christopher Lee 1997

[Soares&Souza,98] HyperProp: Sistema Aberto Hipermidia e Aplicacoes
Luiz fernando G. Osares
Guido L. De Souza 1998

[Gea&Gutierrez,01] Modelado de Sistemas Colaborativos en Base a Estructuras de Naturaleza Hipermedia
Miguel Gea, Francisco Luis Gutiérrez, Jose Luis Garrido
Granada, España, 2001

SITIOS DE INTERNET

URL:
http://www.comp.lancs.ac.uk/computing/research/cseg/projects/ariadne/docs/survey.html

URL: http://education.indiana.edu/~frick/youngkin2.html

URL: http://ksi.cpsc.ucalgary.ca/courses/547-95/yin/groupware.html

URL: http://ksi.cpsc.ucalgary.ca/courses/547-95/pfeifer/cscw_domain.html

URL: http://cs-people.bu.edu/dgd/version.html

URL: http://www.collaborate.com/publications/chapt_toc.html

http://www.collaborate.com/publications/techapp2.html

URL: http://cs-people.bu.edu/dgd/thesis/original_paper.html

URL: http://cs-people.bu.edu/dgd/workshop/whitehead.html

URL: http://cs-people.bu.edu/dgd/workshop/soares.html

URL: http://www.asis.org/Bulletin/Oct-98/webdav.html

URL: http://www.ascusc.org/jcmc/vol2/issue3/chen.html

URL: http://www.ascusc.org/jcmc/vol2/issue3/chen.html

URL: http://www.iac.honeywell.com/Pub/Tech/CM/ClearCaseV2.html

URL: http://www.dat.etsit.upm.es/servicios/cursillo.www/cuarto/html/cvs.html

URL: http://www.gnu.org/manual/cvs/html_mono/cvs.html

URL: http://durak.org/cvswebsites

URL:
http://www-cgi.cs.cmu.edu/afs/cs/project/vaschelp/www/Archiving/Cvs/cvs.html

URL: http://www.cnpq.br/dpe/protem-cc/artigos/hyperprop.htm