Universidad de las Américas

Facultad de Ingeniería de Sistemas

Creación de una Composición Gráfica en Quartz Composer

Trabajo de titulación presentado en conformidad a los requisitos

para obtener el título de Ingeniero de Sistemas

Profesor Guía: Ing. Santiago Albuja

Jaime Andrés Dávila Guerra

2007

AGRADECIMIENTOS

En la vida de un ser humano pasan muchas personas por su camino, suceden eventos que le enseñan cosas nuevas y afianzan su experiencia. Aquellas personas no necesariamente hacen algo positivo por nosotros, a veces no se dan cuenta del valioso aprendizaje que nos han legado y otras veces solo un gesto ha bastado para darnos guía.

Para aquellos que creyeron en mi, os agradezco y les devuelvo en este, un primer proyecto, el resultado de su confianza. Para aquellos que no depositaron su confianza, les dedico este proyecto porque gracias a ustedes comprendí que cada uno de nosotros es responsable de los éxitos y fracasos que se logren en la vida.

Las siguientes personas, aunque no sean todas, tienen mi mayor agradecimiento por lo que han aportado a mi existencia; el orden no es escencial, lo escencial es cuan agradecido estoy porque ustedes hayan aparecido en mi vida.

David Dávila, Elías Dávalos, Dr. Vallejo, Sasha, Raúl, Alejo, Chori, Andrés Torres, Mauricio Alarcón-Salvador, Juan Carlos Trujillo, Santiago Albuja, Mario Valarezo, Mayra Valle, Xavier Armendariz, Jaime Naranjo, Sheyla, Xavier Luna, Paulina Vaca, Andrés Pinto, Laura Vaca, Esteban González, Danny Cruz, Patricio Guarderas, Alfredo Guerra, Lucrecia Guerra, Patricio Dávila, Diana Dávila, Javier Dávila, Verónica Terán, Jorje Zalles, Janneth …, los niños sordos del instituto.

Como Alicia crucé el espejo, ví maravillas, vi al Jaberwocki y volví a este mundo; a este mundo donde 5 personas son importantes:
Mis padres, mis hermanos y la mujer que amo.

Jaime Andrés Dávila G.

RESUMEN

En el capítulo 1 se encuentra un recorrido histórico no solo del sistema operativo de Apple, también se incluye la evolución tecnológica desde los años 50's hasta UNIX, luego se recorre brevemente su contraparte gratuita: Linux. Por motivos históricos y con el objetivo de desterrar de nuestro medio el mito arraigado al sistema operativo de Apple se realiza un recorrido por la historia de Microsoft desde la aparición de MS-DOS hasta su actual sistema operativo Windows. También se encuentra una breve descripción de los distintos tipos de licenciamiento – GPL, Apple, Microsoft, OpenSource.

En el capítulo 2 se realiza una inducción al sistema operativo OS X: sus capas, sus características, sus habilidades, así como los ambientes de desarrollo existentes, sin embargo antes de centrarnos en el ambiente OS X se realiza un breve estudio del sonido y del color para dejar sentadas las bases de comprensión para la elaboración del proyecto práctico: *Sonido Visual*.

La elaboración del software uitilizando el Proceso Unificado de Rational (cuya breve descripción se encuentra en el capítulo 2) se encuentra en el capítulo 3. Y en el capítulo 4 se encuentran las conclusiones obtenidas de la investigación, desarrollo del proyecto y los resultados de las pruebas realizadas a un grupo de niños sordos.

Tabla de Contenidos

INTRODUCCIÓN

La Universidad de las Américas ha logrado un prestigio por la calidad de educación en la carrera de Ingenieria en Sistemas; sin embargo ha especializado su espectro de enseñanza a la plataforma Microsoft, mientras la investigación de otras plataformas – OS X, Linux, UNIX – ha quedado relegada a un tercer plano y a la iniciativa propia del estudiante.

Mac OS X continúa llevando a cuestas el mito de exclusividad para diseño gráfico, lo cual es falso, ya que el núcleo es UNIX FreeBSD 8.0 y la arquitectura sobre la que se ha desarrollado utiliza tecnología como JAVA, OPENGL, etc; así como metodologías de desarrollo avanzadas; por este motivo el desarrollo de aplicaciones en dicha plataforma puede realizarse con calidad.

El objetivo de este proyecto de tesis es:

i. Creación de un software haciendo uso del ambiente de desarrollo seleccionado – Quartz Composer – y del Proceso Unificado. Dicho software tiene como objetivo generar patrones visuales que correspondan con señales sonoras capturadas a través de un microfono, permitiendo a las personas con discapacidad auditiva visualizar el sonido.

ii. Entregar a la UDLA una documentación detallada del software realizado que permita a los estudiantes implementar mejoras que beneficien a las personas con discapacidades auditivas.

iii. Mostrar otras posibilidades de desarrollo de software: Apple incluye ambientes de desarrollo implementados sobre una arquitectura estable que permite la creación de aplicaciones de calidad, en este proyecto se presentará de forma específica Quartz Composer que es un ambiente de desarrollo para procesamiento de datos gráficos.

1 SITUACION ACTUAL

"Sin el software, una computadora es en escencia

una masa metálica sin utilidad"

Andrew S. Tanenbaum

1.1 HISTORIA DE LA INFORMÁTICA

"...El software para computadoras puede clasificarse en general en dos clases:...programas de sistema..., programas de aplicación..."[1].

De este modo podemos diferenciar aquellos programas que se encargan de controlar el funcionamiento de los dispositivos – hardware – y de los procesos que permiten a un computador trabajar adecuadamente y sobre los cuales pueden instalarse los programas aplicativos.

Los programas del sistema en su conjunto son conocidos como *Sistema Operativo*, el mismo se encarga de mantener un control de los componentes de hardware (memoria, procesadores, relojes, interfaces, etc); elementos críticos para el funcionamiento de los aplicativos que son necesarios para un correcto funcionamiento del sistema, por tal razón "...hace muchos años, quedó claro que debía determinarse una forma de proteger a los programadores de la complejidad del hardware"[2].

1 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1º,1992, pag.1
2 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1º,1992, pag.1

Fig. 1.1 Elementos de un sistema de Computo[3]

El objetivo del sistema operativo es simplificar el trabajo con el computador, tanto a desarrolladores como a usuarios finales, los cuales no deben preocuparse de los detalles del movimiento de cabezas del disco, asignación de recursos para ejecutar un cálculo, etc..., por el contrario, tanto el desarrollador como el usuario final (cada uno en su propio ámbito), se preocupan y se encargan de resolver sus problemas particulares, tales como procesadores de texto, bases de datos, juegos, automatización de procesos, etc...

Para cumplir el objetivo destinado al sistema operativo, el mismo debe estar provisto de una arquitectura adecuada para la funcionalidad que debe desempeñar, por tal razón a medida que los sistemas operativos han evolucionado, su arquitectura también ha mejorado.

3 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1º,1992, Fig. 1-1, pag.2

En los temas siguientes tratarán ciertas características, así como algo de la historia de algunos sistemas operativos; sin embargo, el tema central de este proyecto no consiste en explorar la historia y las características de los sistemas operativos de modo profundo, por tal razón se realizará una revisión de cada uno de los sistemas predominantes en la actualidad y si el caso lo amerita, una exploración un poco exahustiva del tema en cuestión.

1.1.1 Silicon Valley

El término *Silicon Valley* fue acuñado por el periodista Don Hoefler en 1971 y se refiere a Santa Clara Valley, ubicada en San Francisco, antiguamente era conocido como *Valley of Heart's Delight*, debido a que en esos tiempos predominaba la cosecha de frutas, hoy en día su nombre es sinónimo de tecnología y su nombre se debe a la gran concentración de empresas de semiconductores y computadoras que se encuentran en el sitio.[4]

En los años 50 William Shockley, luego de renunciar en *Bell Labs* se mudó a Mountain View y creo *Shockley Semiconductor Laboratory* como parte de *Beckman Instruments*. Lugar en que rediseñó el transistor inventado por él, empezó a experimentar con Silicón (a diferencia de muchos otros investigadores que utilizaban Germanio como material semiconductor). Luego de una disputa legal, ocho de sus más talentosos

4 Wikipedia.org, “Silicon Valley”, http://en.wikipedia.org/wiki/Silicon_Valley, 2006

ingenieros renunciaron y fundaron *Fairchild Semiconductor*, casa natal de lo que luego se llamaría *AMD, National Semiconductor* e *Intel*.

Entre otros acontecimientos históricos sucedidos en Silicon Valley está la influencia en los sistemas operativos, desarrollo de software e interfaz gráfica, por ejemplo: haciendo uso del dinero de la NASA, y de las fuerzas aereas, Doug Engelbart inventó el mouse y la interfaz gráfica en los años 60. *XEROX's Palo Alto Research Center (PARC),* tuvo un rol importante en la programación orientada a objetos, interfaces gráficas para usuarios (GUI), Ethernet, PostScript e impresoras láser.

Las semillas de XEROX llegaron a 3Com, Adobe, CISCO (tenían la necesidad de conectar varios protocolos existentes en la red de Stanford), Apple Computer y Microsoft.

1.1.2 Sistemas Operativos

La historia de los sistemas operativos es reciente, tuvo su inicio en los 40's momento en que se realizaba el *proceso en serie*, que consistía en la interacción directa entre el programador y el hardware, esto volvía lento el proceso, amén de la velocidad de los computadores de aquella época que exigían solicitar turnos de uso del hardware (a través de un formulario de reserva).[5]

5 Wikipedia.org, “Historia de los Sistemas Operativos”,

En los años 50 los sistemas operativos "...hacen una aparición discreta..."[5], con el objetivo de facilitar el desarrollo a los programadores, tales *sistemas operativos* fueron:

- *Monitor residente*, se limitaba a cargar los programas, desde una cinta o tarjeta perforada, a la memoria. Su limitante era el tiempo de retirada de un trabajo y la carga de otro.
- *Procesamiento por lotes*, fue el modo de optimizar el trabajo por cintas, lotes. Consistía en agrupar varios trabajos en un mismo lote y ejecutarlos en secuencia. De este modo se redujo el tiempo de transición entre programas. Se utilizó la técnica de *on-lining*[6]; en un computador periférico, de menor coste, se cargaban los trabajos de cinta perforada a cinta magnética para llevarlos al computador principal.
- *Almacenamiento temporal*, haciendo uso del *buffering(*proceso de usar buffers para retener datos que están siendo movidos de o hacia puertos input/output, como puertos seriales o discos flexibles[7]*)* y del *spooling(Simultaneous Peripheral Operations On-Line,* proceso que almacena durante un breve tiempo los datos que han de ser enviados a una impresora[8]*)* se sincronizaba la terminación de un trabajo con la inicialización del siguiente.

---

http://es.wikipedia.org/wiki/Historia_y_evolución_de_los_sistemas_operativos, 2006

6 Wikipedia.org, "Historia de los Sistemas Operativos", http://es.wikipedia.org/wiki/Historia_y_evolución_de_los_sistemas_operativos, 2006

7 Microsoft TechN et, "Glossary", http://www.microsoft.com/technet/archive/wfw/7_agloss.mspx?mfr=true, 2006

8 Wikipedia.org, "Spooling", http://es.wikipedia.org/wiki/Spooling, 2006

En los años 60's el aparecimiento de nuevos modelos de sistemas operativos permitío incrementar el potencial de los computadores, tales cambios incluyeron:

- *Multiprogramación*, la memoria almacena más de un programa de usuario. Mientras un programa se encuentra realizando una operación E/S, el procesador pasa a ejecutar otro programa. De este modo puede utilizarse optimamente los recursos del procesador.
- *Tiempo compartido*, similar a la multiprogramación; los usuarios trabajan con una terminal en línea y sus programas residen en memoria. Cuando un programa realiza una operación E/S, el sistema cede procesamiento a otro programa (igual a multiprogramación), pero si el programa está demasiado tiempo en ejecución, se lo detiene para que se ejecute otra aplicación. En este modelo, los recursos se reparten entre los usuarios simulando concurrencia.
- *Tiempo real*, se lo utiliza en entornos donde se debe procesar la información en tiempos muy breves. Utilizado en aplicaciones dedicadas, su restricción temporal es bien definida, "...por lo que el procesamiento debe llevarse a cabo dentro de los límites definidos o el sitema fallará."[9]

9 Lidón Garcia, Luis Peralta, Samuel Fernández, "Un Paseo por la Historia", http://spisa.act.uji.es/~peralta/os/

- *Multiprocesador*, son equipos diseñados para trabajar con varios procesadores, los cuales comparten memoria y reloj.

Existieron varios sistemas operativos que nacieron en este periodo, pero debe destacarse el nacimiento de UNIX, sistema operativo que se ha mantenido y ha evolucionado hasta convertirse en un paradigma, situación que se explicará más adelante.

El avance de la electrónica permitió en los años 70's integrar miles de transistores en un centímetro cuadrado de silicio, lo que daría nacimiento a los primeros sistemas integrados.

Para cubrir la brecha producida por los anteriores sistemas operativos, respecto del coste de desarrollo y económico, se realizó un costoso trabajo para colocar una capa de software entre el usuario y la máquina; de este modo el primero no necesitaba conocer nada respecto a la circuitería del hardware.[5]

En este periodo se asienta el prestigio de UNIX (en parte por la aparición de C, que permitío reescribir todo el sistema volviéndolo estable). También aparecen sistemas operativos que fueron abandonados pero sirvieron de base para los sistemas operativos actuales, estos fueron:

- *MULTICS(Multiplexed Information and Computing Service)*, fue uno de los primeros sistemas operativos de tiempo compartido. Desechó la distinción entre archivos y procesos en memoria, implementando un solo sistema de almacenamiento para accesar a los datos. También se destacó por ser uno de los primeros sistemas multiprocesador.
- *MVS (Multiple Virtual Storage)*, fue el sistema operativo más usado por mainframes, para el procesamiento de gran cantidad de datos. A parte de permitir la ejecución de múltiples tareas, introdujo el concepto de memoria virtual, lo que permitía que cada programa tuviera su propio espacio de direccionamiento de memoria.[10]
- *CP/M (Control Program/Monitor)*, fue el sistema operativo más usado por las computadoras personales – Pcs – de esa década. Su éxito radico en su portabilidad (se distribuía en diskettes de 8 pulgadas). Se lo puede considerar el padre de DOS (nacido de QDOS, un clon a corto plazo de CP/M).[11]

A continuación se profundizará en los sistemas operativos de mayor impacto y éxito en el mundo de la informática, de tal forma que se podrá percibir las diferencias entre los distintos sistemas operativos.

10 Wikipedia.org, "Historia de los Sistemas Operativos", http://es.wikipedia.org/wiki/Historia_y_evolución_de_los_sistemas_operativos, 2006
11 Wikipedia.org, "MS-DOS", http://es.wikipedia.org/wiki/MS-DOS, 2006

1.1.2.1 UNIX ®

1.1.2.1.1 Historia

Para entender UNIX primero se debe conocer su historia, evolución y características.

Sus inicios se remontan a 1964, su padre se llamó MULTICS[12] y fue desarrollado por Bell Laboratories, General Electric y el M.I.T. MULTCIS tuvo un fracaso casi inmediato, entre las razones de tal acontecimiento fue el lenguaje de pragramación en que fue escrito, *PL/I.*

Luego del fracaso de MULTICS, Bell Labs renunció al proyecto y uno de los desarrolladores, Ken Thompson[13] (a quien se uniría en un momento posterior Dennis Ritchie), desarrollo un nuevo MULTICS en lenguaje ensamblador en una PDP-7 y luego de probar las funcionalidades y la estabilidad del sistema operativo fue denominado UNICS (*Uniplexed Information and Computing System*) "...de una manera un tanto burlona...para indicar que era un MULTICS castrado..."[14] por Brian Kerninghan.

Cuando UNIX había impresionado lo suficiente se realizaron cambios, primero en el hardware: PDP-7, PDP-11/20, PDP-11/45, PDP-11/70;

12 Descrito en la página 11
13 Sebastían Sánchez, "UNIX Y LINUX. Guía práctica", Alfaomega ,ed. 2, 2003, pag. 3
14 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1, 1992, pag. 300

cabe mencionar que PDP-11/45 y PDP-11/70 tenían "...hardware de protección a la memoria, lo que permitío soportar varios usuarios al mismo tiempo."[15]. Luego se tomó la decisión de modificar el lenguaje de programación sobre el que estaba escrito UNIX: B, un lenguaje de programación cuyas características a pesar de ser amplias no incluían la presencia de estructuras, por lo que Ritchie diseño al sucesor de B, C. La introducción de este nuevo lenguaje en el momento correcto ha permitido que UNIX y C dominen el ambito de sistemas desde sus particularidades: Sistema operativos y lenguaje de programación, respectivamente.

Bell Labs empezó a otorgar licencias a bajo costo a las universidades, las cuales al poseer el código fuente del sistema operativos empezaron a desarrollarlo de mil formas. Se realizaron conferencias, "...en torno e UNIX, con distinguidos conferencistas que indicaban el descubrimiento de cierto oscuro error y la forma de arreglarlo..."[16]. De esta forma, la evolución de UNIX permitío que la versión 6 fuera el primer estándar académico.

Sin embargo existía un inconveniente, el lenguaje C, a pesar, que simplificó el traslado de UNIX a otros computadores implicaba la creación de un compilador C para cada máquina, controladores

15 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1, 1992, pag. 301
16 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1º, pag, 301

específicos y un pequeño código dependiente de la máquina. Por tal motivo "...Steve Jhonson de Bell Labs diseño e implantó un compilador portable de C..."[17], esto produjo varios cambios, se traslado UNIX de la PDP-11 hacia una Interdata 8/32 (El cambio implicaba llevar el código compilado de la PDP-11 a la Interdata, situados en pisos distintos del edificio Bell, por tal motivo nacío la idea de conectar los equipos entre sí, de forma electrónica, de esta forma nace el concepto de redes en UNIX).

Cuando AT&T se liberó de la restricción de monopolio (el gobierno de Estados Unidos tenía una participación en la empresa hasta 1984), liberó un sistema UNIX, *sistema III*. Por su parte, Berkeley que había adquirido la versión 6 de UNIX, con apoyo económico de DARPA (*Agencia de Proyectos de Investigación Especiales de la Defensa*) "...produjo una versión mejorada para la PDP-11, llamada 1BSD"[18]. La aparición de 4BSD tuvo gran impacto, ya que a diferencia de todos aquellos sistemas UNIX, que en escencia eran la versión 7, 4BSD contenía un gran número de mejoras:

- Uso de memoria virtual.
- Presencia de paginación (permite que los programas sean más grandes que la memoria física.).
- Los nombre de los archivos podían ser mayores a 14 caracteres.

17 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1º, pag, 302
18 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1º, pag, 303

- Se modificó la implantación del sistema de archivos.
- El manejo de señales era más confiable.
- La introducción del uso de redes trajó la implementación del protocolo de redes BSD, TCP/IP

Todo lo anterior, unido a la implementación de programas utilitarios como *vi* (editor de texto), *csh* (un nuevo shell), compiladores de Pascal y Lisp, permitío que el UNIX de Berkeley se estableciera con firmeza en el mundo académico (y posteriormente, en el mundo comercial), con la preferencia de Sun Microsystems, DEC y otros que decidieron basar su versión de UNIX en el BSD, en lugar de versión V de AT&T.[19]

1.1.2.1.2 El ambiente UNIX

"UNIX es un sistema interactivo de tiempo compartido...diseñado por programadores para programadores"[20], fue creado para ser sencillo, elegante y consistente (la característica de sencillez ha perdido fuerza a través de la modificaciones y adaptaciones surgidas; actualmente es un sistema complejo e inmenso), potencia, flexibilidad y carencia de redundancia inútil.

19 Para una completa revisión del árbol genealógico de UNX, refierasé, Anexos::Genealogía de UNIX
20 Andrew S. Tanenmbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1, 1992, 306

La arquitectura de UNIX es uno de los elementos que ha permitido a este sistema operativo ser tan popular y sobrevivir a los cambios de tecnología. Tal arquitectura está compuesta por:

- ***Sistema operativo***, que se ejecuta en el hardware, lo controla y proporciona una interfaz de llamadas para todos los programas.
- Debido a la carencia de un modo de escribir una instrucción de señalamiento en C, se proporciona una ***biblioteca*** (la interfaz de la biblioteca es lo que se especifica en el estándar POSIX).
- La interfaz compuesta por los ***programas de utilerias***, es lo que un usuario común conoce como UNIX, sin embargo no tiene casi nada que ver con el sistema operátivo y puede ser reemplazado con facilidad.

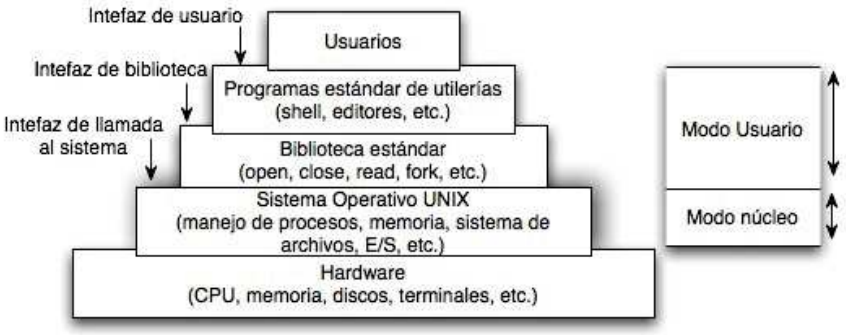


Fig. 1.2 Las capas de un sistema UNIX. [21]

21 Andrew S. Tanenmbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1º, 1992, Fig. 7-2, pag. 307

1.1.2.1.3 Características

La seguridad, el manejo de memoria, las llamadas del shell hacia el sistema operativo, el manejo de archivos y directorios, la implantación, etc... son elementos claves en la comprensión de un sistema UNIX.

- Seguridad: a diferencias de sistemas operativos como MS-DOS (y posteriormente Windows), solo usuarios autorizados pueden acceder a carpetas y archivos, que se encuentran bloqueadas con un alto nivel de encriptación para el resto de usuarios.

  > *El archivo de contraseñas contiene un renglón por usuario, con el nombre de de entrada del usuario, el número de identificación… Al entrar un usuario, el programa login cifra la contraseña recién leída de la terminal y la compara con la otra contraseña...*[22]

- *Shell*: lo interesante de la terminal de comandos de UNIX, que no es más que un programa ordinario para el usuario, es el modo en que se maneja los comandos entre sí, por ejemplo el "entubamiento" – *pipping* – *que direcciona el valor de salida de un comando hacia la entrada de un segundo.*

22 Andrew S. Tanenmbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1, 1992, pag. 308

- *Archivos y Directorios* son considerados una secuencia de bytes, a UNIX no le interesa el significado de los bits, a diferencia del usuario. En un inicio el nombre se restringía a 14 caracteres y BSD aumentó la capacidad a 255.

  Regresando al tema de seguridad, en el contexto de los archivos, cabe mencionar la asignación de 9 bits para determinar los derechos sobre los archivos: propietario, grupo, invitados y la capacidad de lectura, escritura o ejecución.
- *Procesos*: son las únicas entidades activas. Cada uno ejecuta un solo programa con un único hilo de control y el hecho de que UNIX sea multiprogramación, permite ejecutar varios procesos independientes al mismo tiempo.

  Existen ciertos procesos que se ejecutan en el sistema de modo constante y asíncrono, en segundo plano, llamados *demonios*.
- *Memoria:* para poder implantar UNIX en cualquier hardware, cada proceso consta de tres partes:
  - Segmento de texto, contiene las instrucciones de máquina (código ejecutable); el mismo es exclusivo para la lectura, no se modifica ya que esto conlleva complicaciones en la depuración.
  - Segmento de datos, espacio para almacenar datos del

programa.

- Segmento de la pila.

- *Usuarios*: anteriormente se mencionó el modo de protección que tienen los archivos. Cabe mencionar que cada usuario tiene un *uid* (user id), y este se encuentra asociado a la información que puede ver, escribir y ejecutar; sin embargo, existe el *uid 0*, llamado *root*, y es el único con los privilegios totales de acceso, escritura y ejecución de cualquier archivo del sistema.

  Así como hay procesos asignados a cada usuario, el superusuario – root – posee procesos con *uid 0 que* ofrecen prestaciones controladas al resto de usuarios, por ejemplo el programa que informa el espacio en disco.

1.1.2.2 GNU/Linux

Linux es un sistema operativo UNIX-like, desarrollado por Linus Torvald en la universidad de Helsinsky, Finlandia. Es el núcleo – *core* – *de* GNU, desarrollado por Richard Stellman y basado en minix, un pequeño sistema UNIX desarrollado por Andrew S. Tanenbaum.

1.1.2.2.1 Historia

La historia de UNIX[23] está llena de descubrimientos y avances, pero también tiene su lado oculto; las demandas. Debido a que el código fuente es de desarrollo original de AT&T, luego de ser liberado de la restricción comercial, se involucró en el ambiente de desarrollo de software y por consiguiente en la mejora del UNIX desarrollado por Bell Labs y que tomó varios nombres debido a que también fue proyecto de otras áreas de AT&T, el último nombre que tuvo y con el que se lo recuerda es *UNIX System Laboratories* (USL)[24].

USL fue adquirida por SCO (Santa Cruz Operations) y el código fuente es usado como argumento legal contra los mayores vendedores de Linux (IBM, Red Hat, etc…)[25], sin embargo para calmar los temores de prohibiciones mayores en el desarrollo de UNIX, UNIX Lab elaboró un compromiso de *juego limpio* bajo las siguientes normativas:

- Código fuente era lo único que vendía UNIX lab, hasta 1992 cuando fue vendido a Novell y UnixWare aparecío como un producto compilado.
- Publicación de interfaces por parte de AT&T. Contenían una especificación de los puertos disponibles en un sistemas operativo para que este pueda seguir llamandose UNIX. Tanto

23 Refierase a historia de UNIX, pag. 12
24 Christopher Negus, “Linux Bible”, 2006 edition, Willey Publishing, Inc., 2006, pag. 42
25 Christopher Negus, “Linux Bible”, 2006 edition, Willey Publishing, Inc., 2006, pag. 43

POSIX (Portable Operating System) como SVID (System V Interface Definition), patrocinados por BSD y AT&T respectivamente, se han convertido en guías para el desarrollo de Linux.

- Acercamiento Técnico, muchas de las decisiones fueron hechas con base a impactos sobre los desarrollos de otras compañías.

En 1984, luego que varías compañías empezaran a distribuir su propia versión de UNIX y que procesos legales fueran levantados para proteger la integridad del código fuente de UNIX, apareció una organización que sería un camino para Linux: *Free Software Foundation*.

Debido a que el kernel Linux fue la pieza faltante en aparecer en la escena final, fue aquel nombre el que quedaría grabado en la memoria, sin embargo ciertas distribuciones se refieren a si mismas como GNU/Linux. GNU fue diseñado por Richard Stallman como un clon de UNIX, aunque en la actualidad su nombre hace referencia a la licencia de distribución open source.

Linux puede ser visto como como un "*open source UNIX-like*..." "que refleja la combinación de SVID, POSIX y BSD Compliance"[26].

26 Christopher Negus, "Linux Bible", 2006 edition, Willey Publishing, Inc., 2006, pag. 46

1.1.2.2.2 Características

- No existe la obligación de un reinicio constante debido a instalaciones/desinstalaciones de software/dispositivos. Esta característica se debe en gran medida a que Linux, al igual que UNIX, fueron pensados como sistema operativo de servidores.
- No hay interferencia entre servicios, es decir, si se requiere detener o iniciar un servicio, este no afecta otros procesos ni molesta a otros usuarios conectados.
- Software portable, casi sin modificación alguna puede ser implementado en cualquier distribución que se desee; existe software disponible para plataformas como Windows y OS X.
- El usuario es libre de escoger la distribución y cambiar por otra cuando lo desee, los programas pueden ser descargados a través del internet bajo el mismo concepto de libertad.
- Por la razón anterior el usuario puede configurar su sistema operativo a sus necesidades primarias.

1.1.2.3 Microsoft ®

Microsoft nace como una empresa, cuyo fin económico era comercializar "UNIX con el permiso de los laboratorios Bell"[27], sin embargo debido al poco interés, por parte de IBM, en desarrollar un equipo personal y un

27 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1, 1992, pag. 356

sistmema operativo, Microsoft se convirtío en un coloso de la industria informática.

La revisión histórica, así como sus características serán revisadas en orden cronólogico, es decir, la evolución desde MS-DOS hasta Windows.

1.1.2.3.1 MS-DOS

IBM contrató a Philip Estridge para que fuera a Boca Ratón, Florida, y volviera con una computadora personal. Estridge decidió utilizar componentes estándar: intel 8088 con bus de 8 bits[28]. Por otro lado, IBM acudío a Bill Gates (fundador y actual CEO honorífico de Microsoft), quien había construido una versión de BASIC para la Altair (primer *PC*, producido en 1975), para adquirir los derechos sobre BASIC y solicitarle un sistema operativo.

Bill Gates adquirio los derechos sobre 86-DOS: un sistema operativo tipo CP/M (el mismo fue la primera opción de IBM, luego de una sugerencia de Gates[29]) desarrollado por Tim Paterson para Seattle Computer Products. Luego de ciertas mejoras fue llamado MS-DOS – Microsoft Disk Operative System – y estuvo listo para ser incluido en el lanzamiento de la PC de IBM.

28 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1, 1992, pag. 356
29 IDEM, pag. 357

La popularidad de MS-DOS se debe casi exclusivamente a IBM, quien anunciaba con agrado la habilidad de ejecutar la mayor parte del software que se utilizaba en el 8080 con CP/M, además, la mayor contribución a MS-DOS fue hacer a la PC un sistema abierto (listados de ROM, diagramas eléctricos, etc…), esto permitio el aparecimiento de clones y Microsoft aprovechó firmando contratos de exclusividad con varias marcas, tales contratos fueron probados como ilegales, tiempo después en Estados Unidos[30].

> *Si alguien hubiese pensado que después de 10 años, este pequeño sistema, elegido casi por accidente, iba a controlar 50 millones de computadoras, todo se hubiera meditado un poco.*[31]

Hay que notar que el objetivo de IBM con la PC era el de una *consola de videojuegos*, la frecuencia del reloj se la hizo compatible con la frecuencia de los televisores de color de Estados Unidos[32].

Han existido varias versiones de MS-DOS 1.0 hasta 6.0, algunas incluyeron características de UNIX:

- 5.0, liberado en 1991, incluye mejoras en la administración de la memoria, soporte de macros. Aunque existió una versión 6, la versión 5 fue la que dío paso a Windows, donde la idea era que si

30 Wikipedia, "MS-DOS", Wikipedia.org, 2006, http://es.wikipedia.org/wiki/ms-dos
31 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1, 1992, pag. 358
32 IDEM

no podían desechar MS-DOS, "...al menos podemos esconderlo."[33] Sin Embargo se cree que hasta Windows 95, Windows no era más que un shell de MS-DOS, incluso, "...hay expertos que alargan está transformación hasta el Windows NT – Windows XP."[34]

1.1.2.3.2 Windows

Windows nace como una interfaz gráfica para DOS, fue llamada *Interface Manager*, pero Rowland Hanson (jefe de marketing de Microsoft) sugirío *Windows* como un nombre más agradable para los usuarios[35]. La versión 1.0 fue lanzada el 20 de noviembre de 1985 y carecía de funcionalidad por lo que tuvo poca acogida. A la poca popularidad que obtuvo se le debe añadir las disputas legales que inicio Apple, apelando que ciertas funcionalidades eran propiedad del sistema operativo de Macintosh: el basurero y la sobreposición de las pantallas, es decir, las ventanas no podían ser ocultadas por otras.

En 1987 la versión 2.0 tuvo un ligero éxito debido a la introducción de aplicativos como Excel y Word, los mismos eran ejecutados desde DOS y mantenían abierto Windows mientras no se cierren los programas. En ese momento aparece una versión para Windows de Aldus PageMaker

33 Andrew S. Tanenbaum, "Sistemas Operativos Modernos", Prentice Hall, ed. 1, 1992, pag. 362
34 Wikipedia, "MS-DOS", Wikipedia.org, 2006, http://es.wikipedia.org/wiki/ms-dos
35 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-microsoft-windows

(hasta aquel instante solo corría sobre plataforma Macintosh) y, según ciertos historiadores, este es el inicio del éxito de Windows.[36]

La introducción de memoria virtual se dío con la aparición de la versión 3.0 (hasta este momento Apple había apelado como una copia del *look and feel* de su sistema operativo). Esto permitío hacer mejor uso de las habilidades multitareas de DOS. Fue compatible con cualquier procesador Intel de la familia 8086/8088, 80286 y 80386, pero los aplicativos debían ser compilados en modo de 16 bits restando habilidades al procesador 80386 diseñado para 32 bits.[37]

Cuando IBM y Microsoft decidieron finalizar su acuerdo de trabajar juntos en el sistema operativo OS/2, Microsoft cambió el nombre de su versión, OS/2 3.0 a Windows NT. Debido a disputas legales sobre la pertenencia del código de OS/2 Microsoft reescribio, desde cero, Windows NT.

Cuando aparecío la *WfW* – Windows for Workgroups – 3.1 y 3.11, las mismas que mejoraron las prestaciones de red, la API (Application Programming Interface) de Windows se volvío el estándar *defacto* de los consumidores de software.[38]

36 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-microsoft-windows
37 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-microsoft-windows
38 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006,

Uno de los mayores avances de Windows NT, fue la introducción de una API de 32 bits, llamada *win32*. También cabe destacar que esta versión fue la primera en introducirse en el mercado de servidores LAN[39] y desde 1993 experimento un *boom* en el ambiente de oficina.

Con Windows 95, Microsoft quiso cambiar todo su código a modo de 32 bits, sin embargo por razones de compatibilidad mantuvo ciertas partes en modo 16 bits – win16 – esto tuvo un impacto en la eficiencia y estabilidad del sistema operativo.[40]

Windows 98 aparecío como un released mejorado de Windows 95, pero demostró ser más estable que su predecesor, daba mejor soporte al sistema de archivos FAT32 y en un movimiento controversial incluyó en el GUI a Internet Explorer: esto condujo a un litigio entre Estados Unidos y Microsoft, donde el primero declaraba un abuso de poder sobre la pertenencia del sistema operativo de mayor distribución en las PC.[41]

Windows 2000 introdujo como características significativas *Active Directory*, un reemplazo completo de *NT 4.0 Windows Server Model*, incluía estándares como DNS, LDAP y Kerberos. Terminal Services fue incluido en todas las versiones.

---

http://en.wikipedia.org/wiki/history-of-microsoft-windows

39 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-microsoft-windows

40 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-microsoft-windows

41 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-microsoft-windows

Windows Me, que llegaría a conocerse como *Mistake Edition*[42], fue concebido como un proyecto a corto plazo entre Windows 98 y Windows XP. Recibío críticas respecto a su estabilidad y un soporte real a DOS.

Windows XP representa la introducción del núcleo de Windows NT en el mercado del consumidor estándar, al reemplazar el modo de 16 bits por el de 32 bits. También es el release de mayor permanencia, desde 2001 hasta 2007 en que fue lanzado Windows Vista.

En los servidores, Windows Server 2003 representó un gran avance de Windows 2000 server ya que introduce mejores características de seguridad, *wizards* para configurar los distintos servicios y la desactivación de otros servicios por razones de estabilidad. Windows Vista fue lanzado en enero de 2007 y se argumenta que aparte de añadir mejoras en la seguridad será más modular que Windows Server 2003.

Windows Vista incluye características mejoradas para la seguridad: un firewall propio, un reemplazo de la filosofía del administrador llamado User Account Control, un ambiente gráfico mejorado – *Windows aero* – y nuevas aplicaciones: Windows Calendar, Windows DVD Maker y *dashboard.*[43]

42 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-microsoft-windows

43 Para una completa revisión del árbol genealógico de UNX, refierasé, Anexos::Genealogía de Windows

1.1.2.4 Apple ®

1.1.2.4.1 Historia: propietary core[44]

Una de las principales características en la historia de Apple Computer es el desarrollo simultáneo de su sistema operativo y su hardware, ya que desde un incio las computadoras desarrolladas por esta empresa fueron concebidas para trabajar con su propio sistema operativo.

El nacimiento de Apple Computer se remonta a 1976, año en que Steve Wosniak se reencuentra con Steve Jobs en una reunión de Homebrew Computer Club, a la cual Wosniak llevó su equipo personal para presumir sobre su habilidad para llevar el diseño de un computador personal con un procesador Motorolla 6800.

Jobs logró vender la idea desarrollada por Wosniak a *The Byte Shop*, una tienda especializada en computadoras, que al ver lo innovador del equipo solicito 50 de los mismos a $500 cada uno. Luego de obtener las piezas a destajo "*los dos Steves*" lograron cumplir su compromiso y entregaron a tiempo la *Apple I*, que entre las características que lo hacían único estaba el hecho de poder conectarse un televisor cualquiera y utilizarlo como display; "...sin embargo, el texto era desplegado a un terrible velocidad de 60 caracteres por segundo..."[45].

44 Desde System 1 hasta Mac OS 9 el núcleo es prácticamente el mismo y es propietario

45 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-apple-inc

También incluía código de arranque en el ROM y gracias a la insistencia de Paul Terrel, Wosniak diseño una interfaz para cassette, en la que se pudiera cargar y guardar programas.

Apple II aparece en la *Feria de Computación de la Costa Oeste*, el 17 de abril de 1977. “El primer día de exhibición, Jobs muestra Apple II a un técnico textil japonés, Mizushima Satoshi, quien se convertiría en el primer distribuidor autorizado de Apple en Japón”[46]. La mejora en la interfaz incluía la habilidad de retener el display en memoría, lo que permitía incluir gráficos y eventualmente color. El éxito de Apple II se dío cuando Jobs obtuvó un préstamo para poder desarrollar un equipo de mejores capacidades y con nuevas habilidades; en conjunto con “Mike” Makkula, obtuvo un préstamo de $250000 y se fundo Apple Computer el 1 de abril de 1976.

El fracaso de *Apple III* fue la obstinada decisión de Jobs de hacer un modelo “elegante”, por lo que se retiro un ventilador del equipo. Muchos equipos se sobrecalentaron, algunos no llegaron a venderse y a pesar del aparecimiento de una versión mejorada el fracaso del equipo ya se había impuesto.

46 Wikipedia, “History of Microsoft Windows”, Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-apple-inc

En los 80's el paradigma era usar ambientes gráficos, programación orientada a objetos y habilidades de red. Jeff Raskin y Bill Atkinso convencieron a Jobs que enfocará la línea de negocio de Apple hacia ese camino. Luego de negociaciones entre Jobs y los desarrolladores de XEROX Palo Alto, se otorgo 3 días de acceso ilimitado dentro de las instalaciones de PARC a 3 desarrolladores de Apple, a cambio de un millón de dólares en *pre-IPO* (Initial Public Offering) de Apple. Jobs decidió encaminar el desarrollo de su sistema operativo hacia un ambiente gráfico y en 1983 aparecío el computador llamdo *Lisa* a un precio de $10000. Aunque *Lisa* estaba adelantada a su época, el precio era exorbitante y Apple falló en su intento por capturar el mercado de negocios.

En el momento en que Lisa estaba en desarrollo, Apple tenía otro proyecto (visto como departamento), llamado *Macintosh*, inicialmente liderado por Jeff Raskin, cuyo objetivo fue la construcción de un equipo con todo lo que un usuario necesite y que esté a la mano. Steve Jobs, quien había sido expulsado del proyecto *Lisa* pronto puso su atención en *Macintosh*.

El primer sistema operativo como tal había aparecido y se llamaba *System Software*, utilizaba el ambiente gráfico desarrollado por PARC y los desarrolladores de SmallTalk, la barra de menús y los menús pop-up

(desarrollados por Apple), el concepto de drag 'n drop (desarrollado por Jeff Raskin) y la iconografía desarrollada por Susan Kare (quien luego diseñaría la iconografía para Microsoft Windows 3.0[47]).

Algo característico en el SO de Apple fue el no-uso de un ambiente a base de línea de comandos, utilizaba enteramente un ambiente gráfico; aparte del kernel se encontraba el *Finder*, un aplicativo usado para manejar los archivos (hasta la actualidad *Finder* realiza dicha tarea y la de mostrar el escritorio). No ejecutaba, sino solo un aplicativo a la vez;. Soportaba el sistema de archivos MFS (Macintosh File System) y de modo incompleto daba soporte a subdirectorios. System 3.0 soporta SCSI y AppleTalk e introduce el concepto de *Trash* (papelera de reciclaje en Windows).

System 5 incluye el *multiFinder*, un reemplazo de *Finder* que permitía correr varios aplicativos al tiempo, bajo el concepto de *co-operative multitasking*[48] los programas en *background* tenían tiempo de procesador cuando el programa en *foreground* abdicaba. Una característica significante en *System 5* fue la introducción de *Color QuickDraw*, que cambiaría el diseño y la arquitectura gráfica – y sus APIs – del sistema de Apple.

47 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/Mac_OS_history
48 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/Mac_OS_History

System 6 es considerado el primer gran upgrade del sistema operativo Mac, era una versión que consolidaba todos los *releases* anteriores en un ambiente más estable.

En 1985, debido a una pugna de poderes entre Jobs y Jhon Sculley (CEO de Apple), la junta directiva pidio a Steve Jobs su renuncia. Jobs compró Pixar y "...fundó NeXT Inc., una compañia que construía máquinas con diseños futuristas y corrian un sistema derivado de UNIX, NeXTStep."[49] Aunque NeXT no tuvo gran éxito comercial, representa un hito conceptual, ya que el sistema operativo actual de Apple, Mac OS X, está basado en NeXT; además cabe mencionar que "...sirvió como plataforma inicial para que Tim Berners realizará el desarrollo del..."[50] "...primer web browser"[51].

System 7, liberado en 1991, ofrecío un ambiente gráfico más colorido, aunque era opcional debido a que ciertos monitores continuaban trabajando en modo monocromático. Introduce los alias, el equivalente de lo que más tarde Microsoft Windows llamaría "Acceso directo"; también hace uso real de los procesadores de 32 bits. AppleScripts, un lenguaje script propio de Apple, así como la estandarización de *True Color* y *TrueType.*

49 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-apple-inc
50 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-apple-inc
51 Amit Singh, "A Brief History of Mac OS X", 2006,http://www.kernelthread.com/mac/osx/history.html

En 1993 es lanzado *Newton*, considerado el primer PDA y aunque comercialmente no tuvo éxito fue "...la inspiración de dispositivos tales como Palm Pilot y Palm PC."[52]. 1994 representó un avance en el ambiente de hardware para Apple: luego que se rompiera la alianza entre Microsoft e IBM, el segundo se alío con Apple para un proyecto llamado *PreP*; en la línea *Power Macintosh* se incluyó el procesador *PowerPC*, que utilizaba una arquitectura tipo RISC.

En 1997, Steve Jobs regresa a Apple como el nuevo CEO (luego que el anterior CEO tuviera el record en pérdidas para Apple) y NeXT es adquirido en conjunto con su sistema operativo que pasaría a ser la piedra angular para el sistema operativo Mac OS X. En este momento Jobs firma un *joint venture* con Bill Gates, tal alianza incluía el compromiso de Microsoft de realizar *releases* de Office (Gates compromete a Microsoft a que Office::Mac tendría características avanzadas antes que Office for Windows) así como la compra de acciones sin derecho a voto por $150 millones, a cambio Apple incluía Internet Explorer como el browser por *default* en su sistema operativo.

Mac OS 8.0 y 9.0 fueron las versiones lanzadas durante este tiempo de cambios en Apple Computer Inc. OS 8.0 introdujo *HFS Plus* (*Hierarchical File System Plus*), un sistema de archivos mejorado. OS

52 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/history-of-apple-inc

9.0 tiene como características notables: el soporte a redes inalambricas 802.11b, un soporte al trabajo multi-usuario, AppleScript presento nuevas habilidades para controlar redes y protocolo TCP/IP. Se incluyeron tecnologías de transición que permitiera a los desarrolladores adoptar ciertas características del anunciado OS X antes que saliera al público, esto permitío que el OS 9.0 pudiera ser portado como un sistema operativo "independiente" dentro de OS X, tal ambiente fue llamado *Classic*.

1.1.2.4.2 OS X

Mac OS X, desde todo punto de vista, es independiente del sistema operativo anterior de Apple, está basado en un núcleo UNIX-like conocido como Darwin[53]. Darwin es un release, bajo licencia *Open Source*, de las capas más internas de Mac OS X: el kernel Mach-O – basado en el microkernel Mach 3.0 – cuyas funciones consisten en:

- Memoria protegida
- Comunicación entre procesos
- Manejo de interrupciones
- Preemptive Multitasking[54]

53 En el ambiente de OS X el proyecto Darwin, visto como kernel, es conocido como XNU: anagrama recursivo inverso de *It's Not UNIX*

54 Amit Singh, "XNU: The Kernel", 2006, http://www.kernelthread.com/mac/osx/arch_xnu.html

El otro elemento incluido en Darwin es la capa FreeBSD, sistema UNIX desarrollado y liberado por la universidad de Berkeley, California. Sus características más distintivas son:

- modelo de procesos
- POSIX API
- BSD sockets, firewall
- Crypto framework
- Mecanismos de sincronización

Debido al gran apogeo de Java, durante este periodo de transición, Apple incluyo una JVM de alta velocidad para soportar Java, así como el hecho de exponer las APIs de COCOA al lenguaje Java.

COCOA es el ambiente de programación para aplicaciones nativas orientadas a objetos. Es el API de mayor uso entre los desarrolladores del ambiente OS X y será tratado con mayor detalle más adelante.

El nacimiento de OS X deriva de OpenSTEP – sistema de libre distribución derivado de NeXTStep – que se incluyó en la adquisición de NeXT el 4 de febrero de 1997. Durante dos años los esfuerzos se concentraron el portar las APIs originales de Macintosh a “...librerias

UNIX conocidas como *Carbon*."[55]

En el ambiente gráfico, Apple evolucionó su *theme* a algo llamado Aqua, aunque será visto en temas posteriores, Aqua incorporá escalabilidad gráfica, anti-aliasing de texto y gráficos, sombras, luces, animación y transparencias. Sin embargo, Apple respeto su propio estándar conocido como Apple Human Interface Guidelines.[56]

Desde 1999 Apple a realizado lanzamientos contínuos de su sistema con varias mejoras. A pesar que muchos de estos cambio han sido considerados como versionamiento el uso del nombre OS X (10.x) se ha mantenido para dennotar el ambiente UNIX que tiene en su núcleo. La primera versión oficial fue "Cheetah" estrenada el 25 de septiembre de 2001, le siguío "Puma", "Jaguar", "Panther", "Tiger" (versión actual con release 9, es decir 10.4.9) y está anunciado para julio de 2007 en la conferencia de desarrolladores de Apple el lanzamiento de "Leopard" - OS X 10.5.

La arquitectura del sistema operativo OS X está compuesto por varias capas y pueden ser comprendidas con mayor facilidad con el siguiente gráfico.

55 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/History_of_Mac_OS_X

56 Wikipedia, "History of Microsoft Windows", Wikipedia.org, 2006, http://en.wikipedia.org/wiki/History_of_Mac_OS_X

Fig. 1.3 Vista superficial de las capas de OS X[57]

El proyecto Darwin incluye muchos de los paquetes de OS X en modo de libre distribución e incluye muchos de los paquetes estándar del mundo Linux, haciendo que Apple maneje una integración entre su sistema propietario (Cocoa, QuickTime, Quartz Composer) y lo correspondiente al Open Source (Kernel, OpenFirmware, BootLoader, BSD, Java).

Cuando arranca el sistema, OpenFirmware inicia; es el equivalente de BIOS en una PC. Luego de varios procesos arranca BootX – el BootLoader de OS X –, cuya característica primaria es que puede cargar el formato de Mach-0 (el microkernel de OS X),.

57 Amit Singh, "Architecture of Mac OS X", 2006, http://www.kernelthread.com/mac/osx/arch.html

Una vez cargado el sistema operativo se activan servicios interactivos tales como:[58]

- *Core Services*
  - *CarbonCore*, servicios del API Carbon que permite a los aplicativos de Mac OS 9 correr nativamente en OS X.
  - *CFNetwork*, API que permite al usuario trabajar con protocolos de red: FTP, HTTP, LDAP, SMTP, etc…
  - *OSServices*, framework – libreria – que contiene APIs para interactuar con el sistema operativo: System Keychain, Sound, Power, etc…
  - SearchKit, framework para búsqueda e indexación de texto en múltiples lenguajes.
  - *WebServicesCore*, APIs para trabajar con SOAP, XML-RPC.
  - *CoreFoundation.framework,* Provee una colección extra de servicios, por ejemplo permite que aplicaciones accedan a *property lists, URLs,* etc…
- *Application Services*
  - *Quartz Compositor,* consiste de *Window Server* y de librerias privadas. *Quartz* implementa un motor de composición, en el que cada pixel puede ser compartido en tiempo real.
  - *Quartz 2D*, librería para renderizar grápicos en 2D.

58 La descripción de las distintas capas de servicio son extraidas de: Amit Singh, "Above de Kernel", 2006, http://www.kernelthread.com/mac/osx/arch._syshtml

- *Rendering libraries*, OpenGL (2D y 3D), QuickDraw (2D), QuickTime.
- *Other frameworks*, Apple Events, Apple Type Services, ColorSync, CoreGraphics, FindByContent, HIServices, LangAnalysis, LaunchServices, PritnCore, QD, SpeechSynthesis.

• *Application Enviroment* (Ambiente de Aplicaciones)

- *BSD*, similar al tradicional BSD y provee una API POSIX BSD-based.
- *Carbon*, conjunto de APIs para trabajar en ambiente Carbon (explicado anteriormente).
- *Classic*, tecnicamente es un virtualizador que ejecuta todo el ambiente operativo de Mac OS 9.
- *Cocoa*, API orientado a objetos para aplicaciones desarrolladas en Objective-C y Java. Representa una gran legado de NextStep (luego llamado OpenStep). Como se indicó anteriormente, este ambiente es el objetivo de estudio de la investigación actual por lo que se verá con más detalle en capítulos posteriores.
- *Java*, ambiente que consiste de un JDK, clases de Java.

1.2 LICENCIAMIENTO[59]

El modo en que se distribuye el software depende de la licencia adjuntada, la licencia privativa más conocida es EULA – End User License Agreement – de Microsoft, sin embargo existen muchas más: Apple, IBM, Sun MicroSystems, etc… Por el contrario, la licencia OpenSource más conocida es la *GNU General Public License*, GPL, iniciada por Richard Stellman y que busca garantizar la libertad para transmitir el código fuente, así como la libertad para modificarlo y redistribuirlo bajo las mismas condiciones.

Una de las entidades reconocidas para el seguimiento de las licencias OpenSource es *Open Source Initiative*, OSI, que promueve el concepto de compartición y libertad.

1.2.1 GNU

Está licencia busca asegurar una libertad de transmisión y modificación del software adquirido y licenciado bajo GNU. Tal libertad se obtiene al incluir en cada distribución del software una copia de la licencia GNU, también exige que el desarrollador incluya comentarios en las áreas donde modificó el código y que respete la autoría de los desarrolladores iniciales incluyendo los créditos del desarrollo.

59 Refierase a la carpeta /Bibliografia_pdf/Licenses para una copia de las licencias tratadas en este tema o www.open-source.org

1.2.2 BSD & FreeBSD

Las dos licencias manejan un concepto similar: la obligatoriedad de incluir una copia de la licencia en la que certifica que el código fuente es copyright de la Universidad de Berkely, California.

Sin embargo la licencia BSD incluye dos items que exigen la inclusión de un texto certificando que tal producto incluye software desarrollado por la Universidad de Berkeley, California y la clara indicación de no patrocinio por parte de la Universidad de Berkeley a dicho producto sin la autorización por escrito.

1.2.3 EULA – Microsoft

La licencia de Microsoft en escencia es siempre la misma; sin embargo, de acuerdo al producto o servicio licenciado pueden existir claúsulas extras, las mismas suelen incluir una descripción del software o las prestaciones del servicio. Lo que EULA hace es dejar en claro que cualquier intento por redistribuir el software será penalizado, la ingeniería reversa (proceso para obtener el código fuente de una aplicación) es totalmente prohibida y la copia del software puede ser hecha unicamente con fines de respaldo.

1.2.4 Apple

Apple tiene una licencia privativa y una licencia OpenSource, es decir: productos como el sistema operativo, Mac OS X, aplicativos, iLife, iWork, etc, tienen una licencia que permite el uso del software sin permiso de distribución, modificación del código fuente (ingeniería reversa) o copia del paquete, cualquier contraversión a estos lineamientos incurren en acciones legales por parte de Apple.

Respecto a la licencia OpenSource de Apple, existen restricciones en el copyright: se debe incluir en el código modificado los comentarios que denoten los derechos de autor.

Respecto al uso de la marca, Apple extiende *Apple Logo License*, que permite a los desarrolladores de OS X incluir en sus paquetes el logotipo de apple, con previa autorización escrita, como indicativo que su aplicación está desarrollada para trabajar en el ambiente OS X.

1.3 APPLE EN LA ACTUALIDAD

Desde sus inicios a Apple se lo consideró una empresa que se encuentra a la vanguardía de la tecnología, son los inventores de *FireWire*[60], fueron los primeros en implementar un ambiente gráfico para

60 Estándar para transferencia de datos multimedia que funciona 3 a 4 veces más rápido que USB2.0

el usuario. En cada uno de sus productos implementan algo novedoso (*Apple TV*[61] incluye el nuevo protocolo 802.11n) o incluso diseñan productos existentes con características sorprendentes: *iPhone* es un celular cuyo sistema operativo es OS X e incluye capacidad para conectarse a redes inalámbricas y funciona en su totalidad al tacto.

Comenzando con la version OS X 10.1 cada versión ha incluido características dignas de apreciarse: 1) Spotlight, buscador de contenido capaz de leer cada documento para encontrar lo que el usuario necesita. 2) Automator, ambiente gráfico para facilitar la automatización de procesos basandosé en AppleScript.

Apple incluye en cada una de sus versiones utilidades de asistencia para discapacitados – *Assistive Technologies* – facilitando la interacción entre el usuario y el sistema: 1) habilidad de lectura de cualquier documento, 2) respuesta a comandos hablados, 3) modificación de resolución y color de los monitores para usuario daltónicos, entre otros.

Para finalizar esté capítulo se dará una posible respuesta a la pregunta que más se escucha en el ambiente de sistemas operativos; ¿Por qué Apple? La respuesta no es sencilla, existen ambientes con prestaciones similares e incluso mejores tanto en GNU/Linux, UNIX o Windows. La

61 Dispositivo inalámbrico para transmitir multimedia del computador al televisor

seguridad es tratada con la misma importancia en GNU/Linux además de ser un sistema de libre distribución; la existencia de software es más variada en Microsoft Windows y menos del 10% de la población utiliza OS X[62].

La facilidad de uso, las prestaciones gráficas, el ambiente de desarrollo, entre otras características es lo que hace a OS X un sistema operativo excelente. Así que la respuesta a la pregunta es ¿Por qué no?.

62 http://marketshare.hitslink.com/report.aspx?qprid=2 reporta un 6.3% de usuario Mac vs. 91.51% de usuario Windows (Windows98 – Windows XP)

2 MARCO TEÓRICO

“Hay un lapso de tiempo prolongado entre la emergencia del nuevo conocimiento y su destilación en tecnología utilizable”

Peter F. Drucker

2.1 ACÚSTICA

"La Acústica es la disciplina que se ocupa de estudiar el sonido en sus diversos aspectos."[63]

2.1.1 Qué es el sonido?

Es una perturbación sobre un ambiente cualquiera (aire, agua, etc…) que puede ser vista desde un punto de vista físico que explique los fenómenos causa-consecuencia de tal perturbación.

Otro modo de explicarlo es la percepción que tiene un ser vivo – para este estudio se ha considerado al ser humano – respecto del sonido y la relación con el ambiente.

Debido a que existen otras formas de describir el sonido y muchas derivaciones de cada una o en conjunto, en este documento solo se describirán los puntos anteriores y un aspecto importante: *La analogía entre las pulsaciones acústicas y las pulsaciones eléctricas*.

2.1.1.1 Propiedades del sonido

La perturbación que se produce sobre un medio cualquiera (el tema se enfoca en el aire) sin importar el agente inicial, genera valores de salida que contiene la información física del sonido; tales propiedades son:

63 Federico Miyara, "Acústica y Sistemas de Sonido", 4º edición, Fundación Decibel, 2004, pag. 1

- *Velocidad del sonido*, representada con la letra *c,* en el aire equivale a 345 m/s y se define como la distancia que se aleja la perturbación de su centro en un tiempo determinado. Se debe tener en cuenta dos aspectos fundamentales: 1) La velocidad no es constante varía con la temperatura del ambiente y 2)" El sonido es independiente de la intensidad de la perturbación."[64]
- *Longitud de onda*, λ (lambda) representa la distancia entre dos perturbaciones sucesivas en el espacio que tiene un espacio variable de 2 cm a 17 m para los sonidos audibles (agudos y graves respectivamente).

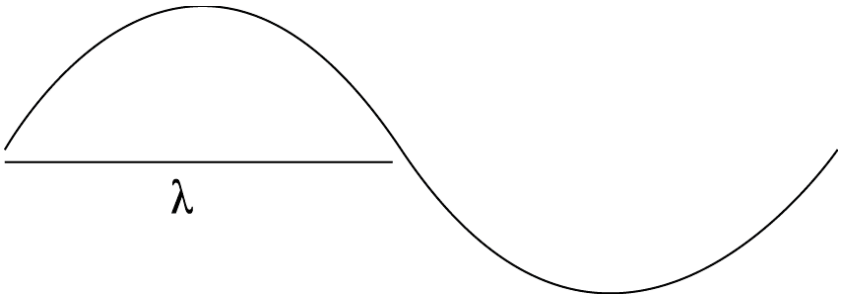


Fig. 2.1 Longitud de onda sonora

- *Periodo*, representado por T y define el "tiempo transcurrido entre una perturbación y la siguiente "[65] que puede variar entre 0.05 ms – agudos – y 50 ms – graves – para los sonidos audibles.
- *Frecuencia*, *ƒ* es un parámetro fundamental en acústica y representa el número de ciclos por segundo, representado en Hz,

64 Federico Miyara, "Acústica y Sistemas de Sonido", 4º edición, Fundación Decibel, 2004, pag. 3
65 Federico Miyara, "Acústica y Sistemas de Sonido", 4º edición, Fundación Decibel, 2004, pag. 6

hertz, varía de 20 Hz – graves – a 20 Khz – agudos.

Una relación importante se da entre el periodo y la frecuencia:

$$f = 1/T;$$

y entre la frecuencia y la longitud de onda:

$$\lambda = c/f$$

- *Amplitud*, representado por P es el máximo valor que alcanza una oscilación en un ciclo, es decir la presión máxima – o valor pico – en un ciclo.

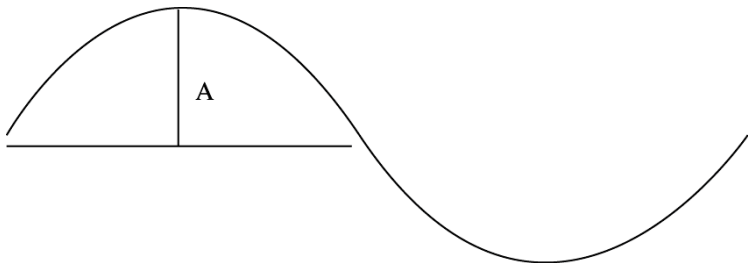


Fig. 2.2 Amplitud sonora

- *Presión sonora*; debido a que las variaciones de presión producidas por el sonido en la atmósfera son muy pequeños, se considera su valor de forma absoluta, por ejemplo si la presión atmosférica en presencia de un sonido es de 100008 Pa (Pa = pascales), la presión sonora es de 8Pa. Basados en el dato anterior se determinó que los valores audibles varían entre 0.00002 Pa – µPa – y 20 Pa.[66]

66 Federico Miyara, "Acústica y Sistemas de Sonido", 4º edición, Fundación Decibel, 2004, pag. 7

Debido a que los valores definidos en Pa son demasiado pequeños para manejar se ha definido una *presión de referencia* que equivale al valor audible más pequeño (0.00002 Pa) y mediante la fórmula,

$$NPS = 20 \log_{10}(P/Pref)$$

se estableció la unidad *Decibel*.

- *Espectro del sonido*, constituye la información de cada armónico con su frecuencia y amplitud respectiva. Un armónico es un sonido con onda senoidal.[67]

  Fisicamente la onda senoidal es la más sencilla – a pesar que en las matemáticas posee un alto grado de complejidad y "está representada por la función trigonométrica *seno*"[68] – pocos sistemas poseen una simplicidad como para ser descritas como senoides. La simpleza de la senoide radica en consta de una sola frecuencia.

2.1.1.2 Percepción del sonido

Así como en el estudio físico del sonido existen propiedades propias de tal descripción, en el campo de la percepción – el modo en que el oído y el cerebro procesan el sonido – también existen parámetros que dependen en distintos grados de las valoraciones registradas para los

67 Federico Miyara, "Acústica y Sistemas de Sonido", 4º edición, Fundación Decibel, 2004, pag. 13
68 Federico Miyara, "Acústica y Sistemas de Sonido", 4º edición, Fundación Decibel, 2004 pag. 13

componentes de la acústica física.

- *La altura*, permite diferenciar los sonidos de una escala musical y su valoración de agudos y graves viene determinada por la frecuencia; debe aclararse que las relaciones no son directas y por tanto la altura también se ve influenciada por la intensidad del sonido e incluso por el timbre, propiedad vista en detalle más adelante.

  Existe un modo de escalamiento que permite establecer las frecuencias de las notas musicales. Partiendo de **LA** central se multiplica por $\sqrt[12]{2}\bullet f$LA para las notas agudas y se divide para tal valor para obtener las notas graves.

- *Sonoridad*, interpretada como la fuerza, el volumen o intensidad del sonido; está relacionada, en principio, con la *amplitud* (P), sin embargo se ve influenciada por la frecuencia con la siguiente relación:

  si $f_1 = f_2$

  sonoridad = max(P_1, P_2);

  si $f_1 > f_2$

  sonoridad = P_1;

- *Timbre*, es una cualidad muy compleja que ha sido visto desde dos puntos, fundamentales, de vista y que no ha podido definirse con exactitud a menos que se conjuguen los factores a continuación descritos:
  i. El primer enfoque distingue un sonido grave de uno agudo del mismo instrumento: intervienen el espectro y las envolventes. La envolvente primaria está relacionada con las familias de instrumentos mientras que "…las envolventes secundarias dependen de la manera en que se amortiguan las diferentes frecuencias del espectro."[69]
  ii. El segundo enfoque busca determinar las características comunes de todos los sonidos de un instrumento y que a la vez diferencian a este último de cualquier otro. El elemento fundamental es la *resonancia* que "…filtra el sonido favoreciendo determinadas frecuencias más que otras."[70]

- Direccionalidad, define los fenómenos ligados a la asociación que hace el cerebro respecto al lugar de procedencia de un sonido.
  i. El primero de los fenómenos refiere a la diferencia de tiempos de llegada de la onda sonora a los oídos – diferencia no mayor a 0.6 ms.

69 Federico Miyara, "Acústica y Sistemas de Sonido", 4º edición, Fundación Decibel, 2004 pag. 23
70 Federico Miyara, "Acústica y Sistemas de Sonido", 4º edición, Fundación Decibel, 2004 pag. 24

ii. El segundo fenómeno se concentra sobre las distintos valores de presión sonora que son ejercidos sobre el tímpano; tiene dos causales:

1) Distancia entre la fuente y el oido receptor,
2) El efecto de resistencia que ofrece la cabeza (es el velocidad de transmisión del sonido en el hueso, cavidades óseas y cerebro). Por último, las diferencias existentes en el espectro que cada timpano recibe. “En efecto, las frecuencias bajas tienen mayor longitud de onda que las frecuencias altas… Estas alteraciones son más notorias en el oído menos expuesto”.

- *Espacialidad*, define la interpretación que dará el cerebro del espacio circudante entre la fuente de sonido y el receptor. Tal interpretación se realiza luego de percibir los siguientes fenómenos definidos por la teoría acústica:
    - La distancia entre la fuente sonora y el receptor. El cerebro asocia sonidos familiares con la presión percibida en el oído para determinar la distancia. Si el sonido es desconocido, el cerebro lo asocia con lo más parecido a algún registro sonoro.

- Las reflexiones tempranas, que son los primeros “rebotes” sonoros que produce una onda sonora en un espacio cerrado (en campo abierto la onda sonora se propaga hasta volverse inaudible), permite al cerebro crear una aproximada “imagen auditiva” del tamaño que tiene el ambiente en el que se encuentra – sensación llamada **ambiencia**[71].
- La reverberación se produce por la superposición de las reflexiones tardías del sonido; esto permite al cerebro elaborar una clara “imagen auditiva” del entorno “…que puede y debe ser aprovechada en audiotécnica para evocar ambientes de gran realismo.”[72].
- El *efecto Doppler*, que describe la movilidad de una fuente sonora por sus variaciones de frecuencia – en conjunto con la direccionalidad –, no suele ser utilizado en música ya que se supone que los movimientos que realizan los instrumentos musicales (si es que hay tales movimientos) son imperceptibles.

- *Enmascaramiento*, propiedad del oído y no del sonido que se define como la ocultación de un sonido por la presencia de otro. La habilidad del oído para enmascarar sonidos permite al cerebro

71 Federico Miyara, “Acústica y Sistemas de Sonido”, 4º edición, Fundación Decibel, 2004 pag. 27
72 Federico Miyara, “Acústica y Sistemas de Sonido”, 4º edición, Fundación Decibel, 2004 pag. 28

despreocuparse por información innecesaria. Esta habilidad es utilizada en el campo del audio digital para reducir el peso de los datos de la composición al eliminar tonos que resultan inaudibles al oído humano.

2.1.1.3 Analogía acústica

El micrófono convierte la señal audible – el sonido – en una señal eléctrica, y a pesar que se espera que las magnitudes de las dos señales sean idénticas (con un cambio de unidad de medida) la situación es distinta, la conversión es por *analogía*, es decir: la señal sonora se considera una presión sonora mientras la señal eléctrica es evaluada como tensión.

La conversión analógica que se da entre las dos señales debería ser similar, excepto por un cambio de magnitud, sin embargo la presencia de *distorciones* y de *ruido* que se agregan a la señal principal varían la tensión que representa analógicamente a la presión sonora.

- Distorsión, es la deformación de una onda y tiende a producirse en los límites que un dispositivo puede manejar (en cuyo caso se dice que tal dispositivo se saturó)[73]. Existen dos tipos de distorciones no lineales (producidas cuando las amplitudes son grandes):

73 Federico Miyara, "Acústica y Sistemas de Sonido", 4º edición, Fundación Decibel, 2004 pag. 68

i. *armónica*, variación de la forma de onda pero no de la frecuencia fundamental

ii. *intermodulación*, variación de la onda sonora cuando las fundamentales no son senoides puros.

- Ruido, “toda señal espuria o indeseada que se superponga a la señal útil”[74]. Existen dos tipos de ruido

    i. Ruído Acústico, producido por el ambiente circundante que genera ondas sonoras que se superponen a la onda principal.

    ii. Ruído Eléctrico, producido por los componentes internos del sistema de audio (parlantes, amplificadores, micrófonos) aunque no es posible eliminarlo por completo es factible reducirlo al mínimo.

74 Federico Miyara, “Acústica y Sistemas de Sonido”, 4º edición, Fundación Decibel, 2004 pag. 64

2.2 COLOR

El color se deriva del espectro de luz y es percibido por el humano luego que tal espectro interactua con los receptores fotosensibles del ojo.

2.2.1 Propiedades de la luz

La radiación electromagnética se caracteriza por la longitud de onda o su frecuencia (medida en nanómetros y en teraHertz respectivamente) y su intensidad.

El espectro de colores se define como la distribución de la luz a cierta intensidad en una determinada longitud de onda[75]

- *Longitud de Onda*, definida en nanómetros se describe como la distancia entre dos perturbaciones, en este caso específico, la luz

2.2.2 Percepción cromática

La definición de colores es en mayor grado influenciada por la interpretación que el humano realiza luego que el ojo procesó la reflexión de la luz. Tal interpretación depende de los siguientes factores:

- *Intensidad de la luz*
- *Reflexión de la luz,* el color blanco es la reflexión total de la luz

75 Wikipedia.org, "Color", http://en.wikipedia.org/wiki/Color, 2006

- *Absorción de la luz,* el color negro es la absorción total de la luz

También se considera el aspecto fisiológico, es decir las condiciones físicas del ojo humano:

- *Conos,* existen tres tipos de células, las cuales responden a determinadas longitudes de onda. Los conos *S* o *conos azules* responden a la luz alrededor de los 420 nm. Los conos *L* o *conos rojos* responden a longitudes de onda de 564 nm. Los conos *M* o *conos verdes* responden al espectro comprendido a partir de 534 nm. Este es el motivo por el que el ojo humano percibe la luz de modo tricromático.
- *Bastones*, responsables de la visión monocromática y en cierto grado la percepción de los tonos grises de las sombras.

Como percepción cromática no estándar se encuentra la *sinestesia* que es la “relación” de colores con algun elemento externo: letras o música.

2.3 UNIFIED PROCESS: UP

El *Proceso Unificado* (por sus siglas en inglés UP); puede ser visto como una guía para el desarrollo de software pero su importancia es escencial ya que "…la profesión es aún joven. En consecuencia, los desarrolladores necesitan dirección organizativa…"[76]. Por este motivo Jacobson, Booch, Rumbaugh, junto a otras personas y empresas, que han contribuido a UML (Unified Model Language), incorporaron en un entorno único "… metodologías antes separadas, nos sentimos justificados al llamarlo "Proceso Unificado"."[77].

"Un proceso define *quién* está haciendo *qué*, *cuándo* y *cómo* alcanzar un determinado objetivo."[78]

Para desarrollar un software de calidad o mejorar uno ya existente se deben seguir ciertas directrices que permitirán que el trabajo tanto grupal como individual pueda seguir un proceso de evolución, estabilidad e incluso de reestructuración dentro de parametros definidos, tales como: alcance de la funcionalidad, hardware, tecnología, personal, costos de inversión.

76 "El Proceso Unificado de Desarrollo de Software", Ivar Jacobson, Grady Booch, James Rumbaugh, Pearson Eduación S.A., 1º ed., 2000, Prefacio
77 "El Proceso Unificado de Desarrollo de Software", Ivar Jacobson, Grady Booch, James Rumbaugh, Pearson Eduación S.A., 1º ed., 2000, Prefacio
78 "El Proceso Unificado de Desarrollo de Software", Ivar Jacobson, Grady Booch, James Rumbaugh, Pearson Eduación S.A., 1º ed., 2000, Prefacio

El proceso para desarrollo de software no puede quedar en la teorización debe ser la piedra elemental sobre la que se construye un software que cumpla los requerimientos de aquellos entes que vayan a hacer uso del producto – personas, sistemas. Sin embargo para poder realizar una estructuración del "proceso" se lo debe conocer: en este documento se encuentra esbozada la historicidad y las características fundamentales del *Proceso Unificado*: Dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

2.3.1 Historia del Proceso Unificado

Ericsson

- En 1967 modelaba el sistema entero como un conjunto de bloques interconectados – *componentes.*
- Identificaban los bloques estudiando los casos de negocio – *casos de uso.*
- Se entregaba una "descripción de arquitectura" software
- Creado por Ivar Jacobson es conocido hoy como *desarrollo basado en componenetes*.

Objectory

- En 1987 Ivar Jacobson sale de Ericsson y funda Objectory AB, Estocolmo.

- "La arquitectura que conduce a los desarrolladores e informa a los usuarios comenzó a destacar."[79]
- Los flujos de trabajo se representaron en una serie de modelos.
- El propio Objectory fue visto como un sistema.

Rational

- Rational compra Objectory AB en 1995
- Rational desarrolló, con anterioridad, prácticas de desarrollo de software, la mayoría complementaria a las prácticas contenidas en Objectory.
- Booch en 1996:

  "Un estilo de desarrollo dirigido por la arquitectura es normalmente la mejor aproximación para la creación de la mayoría de los proyectos muy basados en el software."

  "Para que un proyecto orientado a objetos tenga éxito, debe aplicarse un proceso iterativo e incremental."[80]
- Entre 1995 y 1997 aparece *Proceso Objectory de Rational 4.1*.
- Se hizo explicita la arquitectura en forma de una *descripción de la arquitectura*.

79 "El Proceso Unificado de Desarrollo de Software", Ivar Jacobson, Grady Booch, James Rumbaugh, Pearson Eduación S.A., 1º ed., 2000, Prefacio

80 "Object Solutions: Managing the Object-Oriented Project", Grady Booch, Addison-Wesley, 1996 – Tomado de "El Proceso Unificado de Desarrollo de Software", Ivar Jacobson, Grady Booch, James Rumbaugh, Pearson Eduación S.A., 1º ed., 2000, Prefacio

UML

- *Grady Booch y James Rumbaugh unifican sus métodos – método Booch y Object Modelling Technique – en Rational.*
- *Ivar Jacobson se une al equipo, publican la versión 0.9 de UML (Lenguaje Unificado de Modelamiento).*

RUP

- Rational se fusiona o compra otras empresas que aportaron significativamente en el desarrollo del proceso.
- A mediados de 1998 aparece el Proceso Unificado de Rational 5.0

2.3.2 Características

El Proceso Unificado es más que un proceso: "es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software…"[81]

- Está basado en *componentes software* interconectados a través de *interfaces*
- Utiliza UML para preparar los esquemas de un software

Sin embargo la piedra fundamental del Proceso Unificado está compuesta por tres elementos claves:

81 "El Proceso Unificado de Desarrollo de Software", Ivar Jacobson, Grady Booch, James Rumbaugh, Pearson Eduación S.A., 1º ed., 2000, pag. 4

Dirigido por casos de Uso

- Los casos de uso inician y son el hilo conductor del proceso de desarrollo de software.
- "Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante"[82]
- El usuario representa alguien o algo que interactúa con el sistema en desarrollo.
- La funcionalidad, descrita por los casos de uso, se ve resuelta a través de la pregunta ¿Qué debe hacer el sistema por usuario?

Centrado en la arquitectura

- "Se describe mediante diferentes vistas del sistema en construcción."[83] dejando de lado los detalles.
- Considera factores como la plataforma, bloques de construcción reutilizables (ej: Guías de Interfaz Gráfica), requisitos no funcionales.
- Tanto la arquitectura como los casos de uso deben evolucionar en paralelo.

82 "El Proceso Unificado de Desarrollo de Software", Ivar Jacobson, Grady Booch, James Rumbaugh, Pearson Eduación S.A., 1º ed., 2000, pag. 5

83 "El Proceso Unificado de Desarrollo de Software", Ivar Jacobson, Grady Booch, James Rumbaugh, Pearson Eduación S.A., 1º ed., 2000, pag. 5

Iterativo e Incremental

- El objetivo es dividir el proyecto en miniproyectos, los cuales son iterativos.
- El producto de cada miniproyecto es un incremento.
- Se consideran dos factores para implementar una miniproyecto
  - Casos de uso que en común amplían la utilidad del software.
  - El tratamiento de riesgos es decreciente, es decir en una primera instancia se seleccionan los más importantes.

2.3.3 Ciclo de Vida

- La *unidad* de medidad podría considerarse el ciclo, que es una iteración a través de 5 estadíos: Requisitos, Análisis, Diseño, Implementación, Prueba.
- Un software se desarrolla en el transcurso de un tiempo definido, el cual esta divido en 4 fases: Inicio, Elaboración, Construcción, Transición.

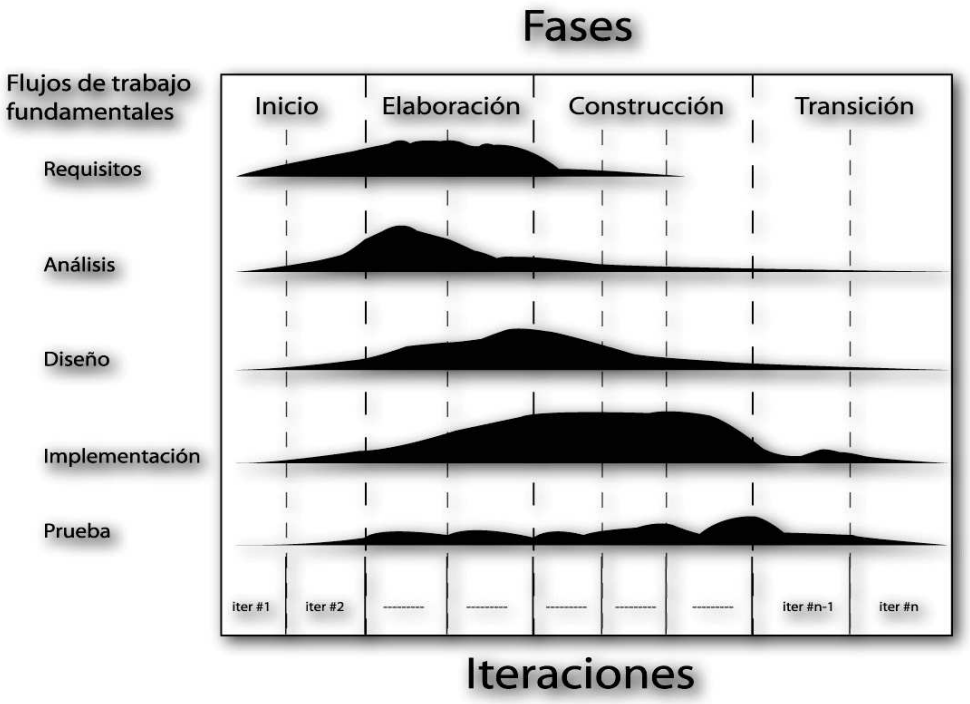


Fig. 2.3 Los cinco flujos de trabajo tienen lugar sobre las cuatro fases [84]

- El producto de cada fase se denomina *hito* y es el compendio de varios *artefactos*: modelos, documentos, prototipos, etc.
- Permiten tomar decisiones cruciales para el proyecto en base a las mediciones obtenidas a través de los artefactos.

84 "El Proceso Unificado de Desarrollo de Software", Ivar Jacobson, Grady Booch, James Rumbaugh, Pearson Eduación S.A., 1º ed., 2000, pag. 11

2.3.4 Producto entregable

- Una de las constantes del desarrollo de software es el cambio de requisitos; tales cambios son propiciados por los cambios del entorno: mejora en sistemas operativos, base de datos, hardware, lenguaje de programación, etc.
- El ciclo de vida del desarrollo de software finaliza con la producción de una versión del sistema.
- Cada versión puede ser vista como un *Producto* entregable que consta de: 1) producto ejecutable, 2) código fuente incluido en componentes listos para compilarse y ejecutarse, 3) manuales, 4) productos asociados.
- En la variedad de productos asociados están incluidos:[85]

    1) Documentación de requisitos
    2) Visión del sistema
    3) Modelo de casos de uso
    4) Especificaciones no funcionales
    5) Modelos de análisis, diseño
    6) Modelos de despliegue, de implementación
    7) Modelo de pruebas

85 "El Proceso Unificado de Desarrollo de Software", Ivar Jacobson, Grady Booch, James Rumbaugh, Pearson Eduación S.A., 1º ed., 2000, pag. 9

2.4 QUARTZ COMPOSER[86]

"Quartz Composer es una herramienta de desarrollo..." incluida en Mac OS X 10.4 "...para procesamiento y renderización de datos gráficos". "Es tecnología a nivel de Sistema Operativo para moción gráfica, diseñado alrededor de OpenGL..."[87]. Originalmente desarrollado por Pierre-Oliver Latour como *Pixel Shox Studio*[88] fue adquirido en 2003 por Apple.[89]

Considerando que el uso de Quartz Composer implica un mínimo de conocimiento respecto del ambiente de desarrollo de OS X, se iniciará con un claro esbozo de la arquitectura sobre la que se fundamentan las distintas APIs, también se dará una breve explicación sobre los Ambientes de aplicaciones . Por motivos teórico referentes a este documento nos centraremos en lo concerniente al tema a tratarse, sin embargo si el caso lo amerita se realizará una breve descripicón y se referirá a la documentación adecuada.

86 Para mayor información: *"Quartz Composer Programming Guide", Apple Refernce Library, developer.apple.com*
87 "Introduction to Quartz Composer", QuartzCompositions, http://www.quartzcompositions.com/phpBB2/mediawiki/index.php/Introduction_to_Quartz_Composer, Last updated: 03/10/2005
88 "Introduction to Quartz Composer", QuartzCompositions, http://www.quartzcompositions.com/phpBB2/mediawiki/index.php/Introduction_to_Quartz_Composer, Last updated: 03/10/2005
89 PixelShox Technology, http://www.pol-online.net/pixelshox_technology/index.php

2.4.1 Arquitectura de OS X

Fig. 2.4 Arquitecura de OS X en vista por Capas [90]

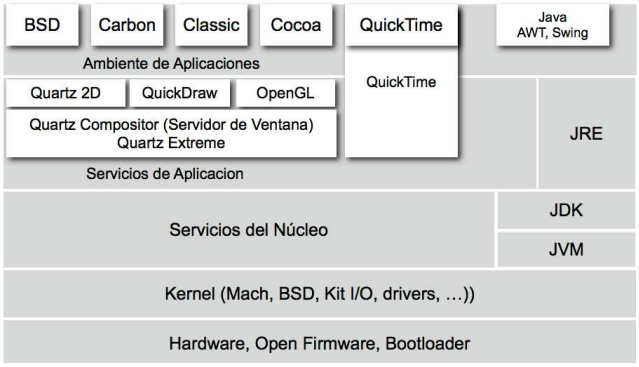


Fig. 2.5 Arquitectura Interna de OS X[91]

- Núcleo de Sistema Operativo: El componente indispensable de OS X es un fundamento basado en UNIX conocido como Darwin, el cual incorpora variadas tecnologías:

90 "Mac OS X Technology Overview", ADC Reference Library, Apple Development Center, 2006, Mac OS X Architectural Overview

91 Amit Singh, "Architecture of Mac OS X", 2006, http://www.kernelthread.com/mac/osx/arch.html

- Mach 3.0 provee funcionalidades transparentes para las aplicaciones como manejo de recursos del procesador, protección de memoria, Soporte de tiempo-real, memoria virtual avanzada (para aplicaciones de 32 bits asigna 4GB de espacio virtual, mientras que para aplicaciones de 64 bits el máximo *teórico* es 18 hexabits).
- FreeBSD 8.0 sirve de base para la implementación de sistemas de archivos, facilidades de red, entre otros.

Otras tecnologías incluídas o soportadas por Darwin incluyen:

- X11, *sistema de ventanas* que permite a muchas aplicaciones UNIX dibujar elementos de la interfaz gráfica.
- Soporte *Dispositio-Driver* a través del framework, orientado a objetos, I/O Kit que permite el desarrollo de drivers con cualidades como: Verdadero *Plug and Play*, Manejo dinámico de dispositivos y administración de carga (portátiles y desktops).
- Soporte Sistema de Archivos, debido a la heterogenidad de los ambientes informáticos, OS X incorpora soporte de varios formatos que incluyen NTFS – *NT File System* –, UFS – *UNIX File System*.

- Soporte de redes, OS X implementa variados protocolos de interconectividad "estándar" (802.1x, TCP/IP, LDAP, etc), *configuración cero – zero Configuration*[92] – y diversas tecnologías para implementar nuevos requerimientos en conectividad.
- Velocity Engine, mejora el rendimiento de cualquier aplicación, haciendo uso del paralelismo de datos se puede obtener lo mejor del procesamiento gráfico, gráfica en 3D, procesamiento de video, compresión de audio.
- Soporte JAVA
- Soporte de 64 bits

- La capa de gráficos y multimedia incluye tecnologías que permiten el desarrollo de ambientes de un modo impresionante y hacen que el trabajo que se realice en estos ambientes sea fluido, eficiente (respecto al procesador) y visualmente soberbio.
  - Quartz, provee soporte para renderizar gráficos 2D y texto y plasmarlo del modo adecuado en el sistema de ventanas (Quartz implica tanto un cliente API – *Quartz 2D* – y un servidor de ventanas – *Quartz Compositor*).[93]

92 Tecnología que habilita el descubrimiento y configuración dinamica de equipos dentro de una red TCP/IP, para mayor información: *"Introduction to Bonjour Overview", Apple Refernce Library, developer.apple.com*

93 Para mayor información: *"Introduction to Quartz 2D Programming Guide", Apple Refernce Library, developer.apple.com*

Quartz 2D incluye características importantes para las aplicaciones de usuario como:

- Renderizado de alta calidad en la pantalla
- Resolución independiente del soporte de interfaz gráfica
- *Anti-aliasing* para gráficos y texto
- Soporte para añadir transparencia a las ventanas
- Generación de PDFs
- Manejo de Color

Quartz Compositor además de administrar el comportamiento del sistema de ventas a bajo nivel, coordina la composición de los elementos visibles en el escritorio del usuario. Soporta efectos de transparencia a través de canales alfa. Incorpora *Quartz Extreme* que hace uso de *OpenGL* y renderiza los gráficos en un hardware especializado, liberando al CPU para otras tareas.

- QuickTime, tecnología multimedia que permite manipular, video, audio, gráficos, texto e incluso realidad virtual de 360º.[94]

94 Para mayor información: *"Introduction to QuickTime Kit Programming Guide", Apple Refernce Library, developer.apple.com*

- OpenGL, estándar de la industria para desarrollo de aplicaciones en 3D. Diseñado para animaciones, juegos, ambientes CAD, OpenGL incluye funciones que permiten el mapeo de texturas, transparencia, generación de efectos ambientales, entre otros.[95]

- Core Image, extiende las habilidades gráficas del sistema al proveer un *framework*[96] para implementar efectos visuales complejos en un aplicación.
    - Corte de imagenes
    - Corrección de color, incluye ajuste de punto blanco (*White-point adjustments*)
    - Variación geométrica de la imagen
    - Control de efectos en tiempo real
    - Efectos lineares: "focos de luz"[97]

- El ambiente de aplicaciones está constituido por ciertas capas definidas por la funcionalidad que aporta al sistema y por el ambiente de desarrollo sobre el que los desarrolladores trabajan.
    - *Carbon*, basado en las interfaces de desarrollo de OS 9.

95 Para mayor información: *"Introduction to OpenGL Programming Guide for Mac OS X", Apple Refernce Library, developer.apple.com*

96 *Framework: Tipo especial de paquete para distribuir recursos compartidos, para mayor información: "Introduction to Framework Programming Guide", Apple Refernce Library, developer.apple.com*

97 Para mayor información: *"Introduction to Core Image Programming Guide", Apple Refernce Library, developer.apple.com*

- *Java*, ambiente multiplataforma.
- *AppleScripts*, permite desarrollar aplicaciones nativos de OS X de modo rápido haciendo uso de *AppleScripts.*[98]
- *Cocoa*, ambiente sobre el que se desarrollan la mayoria de aplicaciones de OS X, ya que incorpora una gran cantidad de comportamientos establecidos por la *guía de interfaz gráfica*[99]*.* El lenguaje de programación utilizado en este ambiente es *Objective-C* u *Objective-C++.*[100]

2.4.2 Ambiente de Desarrollo de OS X

OS X provee ambientes de desarrollo con herramientas diseñadas para volver eficiente la creación de una aplicación. En este capítulo se revisarán brevemente las más involucradas con el proyecto en desarrollo, se aconseja una revisión de la *librería de desarrollo Apple* – ADC.

El más importante de los ambientes para desarrollar aplicaciones por

98 Para mayor información: *"Introduction to AppleScript Studio Programming Guide", Apple Refernce Library, developer.apple.com*
99 Para mayor información: *"Introduction to Apple Human Interface Guidelines", Apple Refernce Library, developer.apple.com*
100Para mayor información: *"Introduction to The Objective-C Programming Lenguage", Apple Refernce Library, developer.apple.com*

haciendo uso de código es *XCode* con un *ambiente integrado de aplicaciones* (*Integrated Developer Enviroment, IDE*) del mismo nombre. Aquí se desarrollan aplicaciones escritas en C, C++, Objective-C, Objective-C++, JAVA, AppleScripts Studio, entre otras. Incluye muchas prestaciones que facilitan y mejoran el desarrollo de software.[101]

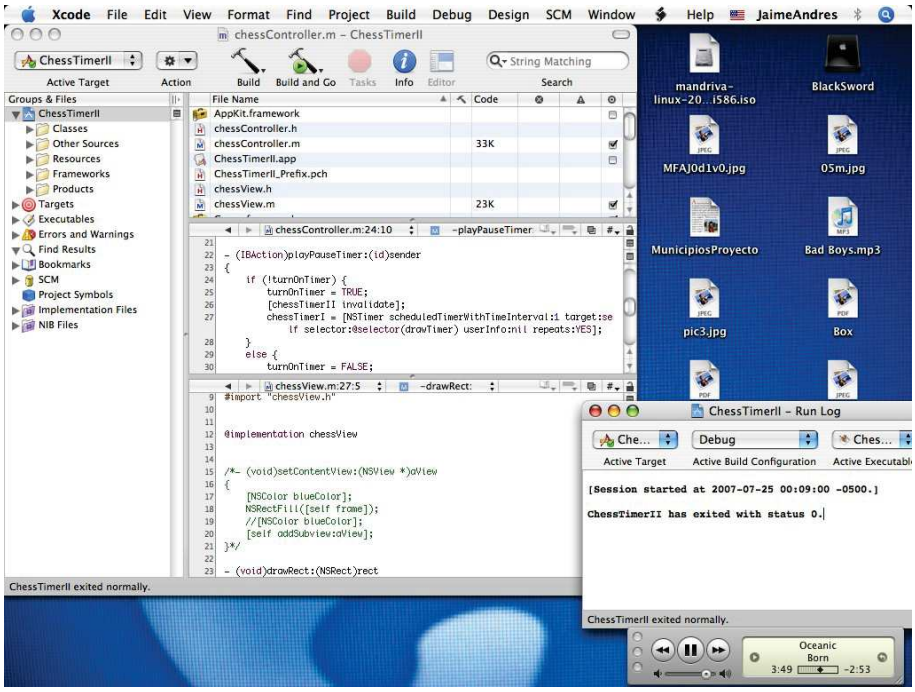


Fig. 2.6 Ambiente de desarrollo Xcode

101 Para mayor información: *"Introduction to XCode 2 User Guide", Apple Refernce Library, developer.apple.com*

Interface Builder es una herramienta para el desarrollo del ambiente visual de la aplicación.

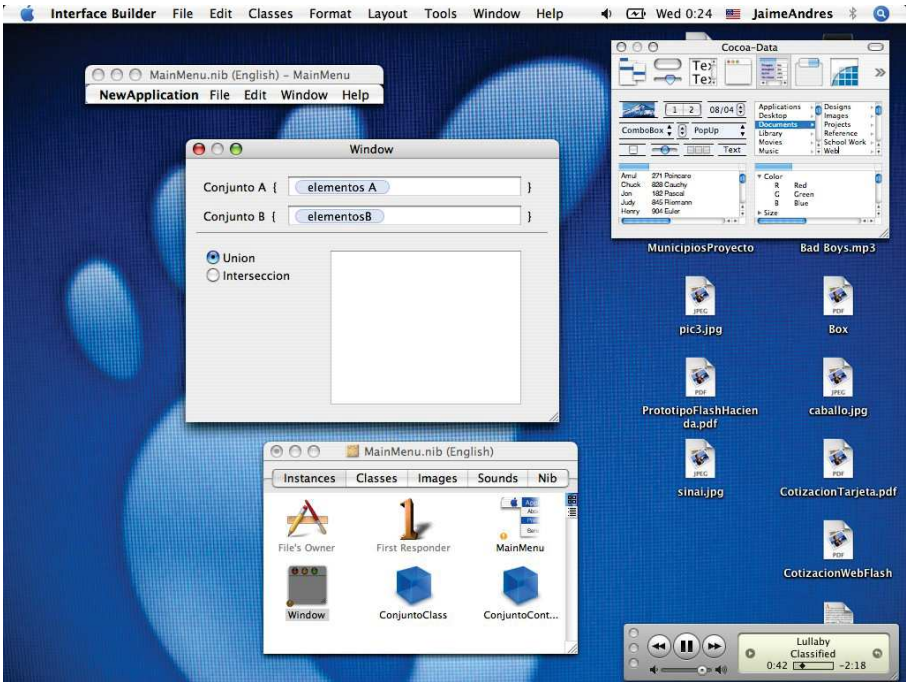


Fig. 2.7 Ambiente Interface Builder

QuartComposer, como se mencionó anteriormente, es la pieza fundamental del proyecto actual. La características que posee lo destaca entre los *Ambientes de Programación Visual* (*Visual Programming Lenguage, VPL*)[102]; tales características permiten crear composiciones visuales, a través de componentes, que pueden funcionar autonomamente o ser incorporadas a otra aplicación.

102 Falta la dirección wikipedia de VPL

El ambiente de trabajo – como se detalla en la figura 2.8 – es amigable, sencillo y consiste de:

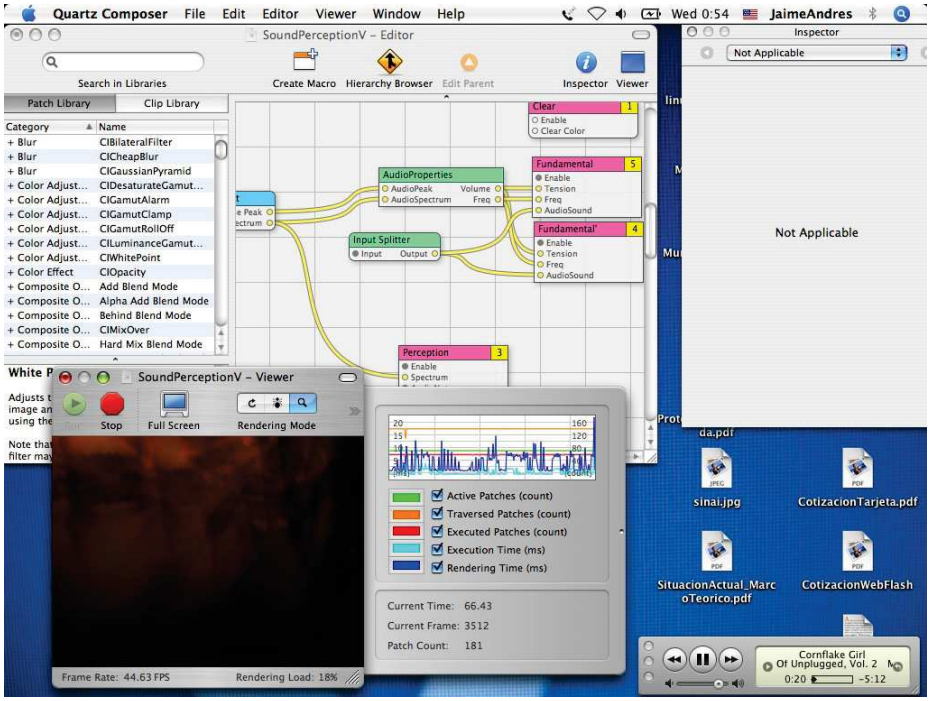


Fig. 2.8 Ambiente Interface Builder

- *Libreria de Parches y de Clips*

  Agrupados por funcionalidad los *parches* constituyen el elemento básico y son representaciones visuales de funciones escritas en Objective-C o C (*framework Quartz.framework)*[103] y filtros para procesamiento de imagen, entregado por *Core Image*.[104]

103Para mayor información: *"Quartz Composer Reference Collection" y "Quartz Composer Programming Guide", Apple Refernce Library, developer.apple.com*

104Para mayor información: *"Core Image Programming Guide" y "Core Image referenceCollection", Apple Refernce Library, developer.apple.com*

Los clips son representaciones visuales de *objetos* que pueden ser incorporados a la libreria para hacer una reutilización colocando la composición en */Library/Application Support/Apple /Developer Tools/Quartz Compser/Clips (o ~/Library/Application Support/Apple /Developer Tools/Quartz Compser/Clips si desea que esté disponible para un solo usuario)*

- *Ambiente de composición*

  Aquí se conectan las funciones a través de puertos y se crean *Macro Parches* que pueden ser considerados como un objeto. También se publican puertos para comunicarse con otros objetos externos dentro de la composición, lo que crea un *árbol de objetos*.

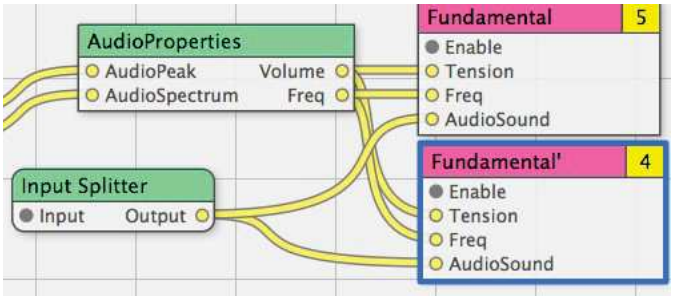


Fig. 2.9 Parches interconectados, los parches pintados de gris están publicados

- *Propiedades*

  Es la ventana donde se modifican las características del parche:

ancho del gráfico, función *javascript* a ejecutarse, etc.

- *Visor de la composición*

  Aquí se renderiza la composición de acuerdo a lo programado.

- *Visor de rendimiento de la composición*

  Existen dos formas de visualizar el rendimiento que hace el procesador con la composición: 1) *Perfil*, entrega estadísticas del uso por parches activos, inactivos, tiempos de renderización y de ejecución y 2) *Debug log*, entrega *logs* del proceso de ejecución y pinta los parches para indicar su estado de ejecución.

3 COMPOSICIÓN QUARTZ COMPOSER

Sonido Visual v2.0

“Los científicos estudian el mundo como es;

los ingenieros crean el mundo que aún no es.”

Theodore von Karman

1 Visión del Sistema

Versión 2.0

Información General

TÍTULO:	Sonido Visual
SUBTÍTULO:	
VERSIÓN:	2.0
ARCHIVO:	Proyecto.odt
AUTOR:	Jaime Dávila
ESTADO:	Prototipo

Firmas y Aprobaciones

ELABORADO POR:	Jaime Dávila Desarrollador de Software		
FECHA:	2006-02-01	Firma:	
REVISADO POR:	Santiago Albuja *Director de proyecto* Mario Valarezo *Revisor de proyecto*		
FECHA:	2006-06-12	Firma:	
APROBADO POR:	Santiago Albuja *Director de proyecto*		
FECHA:	2006-06-12	Firma:	

Introducción

El siguiente documento especifica la construcción de un programa que permita a los sordos visualizar los patrones sonoros generados por una onda de sonido.

Propósito

- Delimitar la construcción del programa para visualizar las ondas de sonido: *Sonido Visual*
- Proporcionar una documentación clara respecto a las prestaciones del producto: *Sonido Visual*

Alcance

- Generación de patrones visuales.
- Controlar “volumen” de salida de los patrones visuales.
- Controlar “equalización” de los patrones visuales.

Definiciones, Acrónimos y Abreviaturas

Remitirse al Glosario de Términos

Referencias

No definido

Posicionamiento del Producto

Definición del Problema

El problema de	convertir el sonido en un patrón visual.
Afecta a	Sordos DJ's Músicos Usuario ordinario
Cuyo impacto es	no disfrute de eventos o productos desarrollados para el usuario ordinario, debido al desconocimiento de la sonoridad.
Una solución exitosa	Un ambiente gráfico que convierta las ondas sonoras a ondas visuales, permitirá a los usuarios asociar ciertos elementos del sonido a la gráfica y al color. Esto permitirá a los sordos involucrarse en un entorno, para ellos, fascinante y novedoso

Posicionamiento del Producto

Para	Usuarios Sordos, DJ's, músicos
Quienes	No tienen una perspectiva visual de su entorno sonoro debido a sus limitaciones físicas; o desean proveer un entorno visual de sus producciones musicales.
Sonido Visual	Es un software de sinestecia
Que	Realiza una analogía entre el sonido y el color facilitando la percepción visual de un entorno sonoro
A Diferencia	de los oscilográmas que entregan las propiedades físicas del sonido para ser interpetradas por gente con el conocimiento adecuado. Así como de las composiciones visuales exclusivas para DJ's que solo poseen información sobre la amplitud y añaden efectos visuales de entretenimiento.

Esta Aplicación	Permite a la gente sorda realizar interpretaciones no solo sobre la amplitud, sino sobre el timbre de un sonido con sus respectivas perturbaciones sonoras. A parte de generar efectos visuales que correspondan con la música de los DJ's, permite controlar la forma en que tal analogía visual se presenta. A los músicos les permite tener una evaluación visual de sus composiciones, lo que les ayuda a perfeccionar su música.

Descripciones de Afectados y Usuarios

Resumen de los Afectados

Nombre	Descripción	Responsabilidades
Sordos	Personas con deficiencia auditiva que no pueden percibir las perturbaciones sonoras de su entorno	Uso directo del software Define los requerimientos funcionales del software
DJ's	Profesionales que realizan mezclas de música a las que incorporan efectos visuales para sus presentaciones en vivo.	Uso directo del software Define los requerimientos funcionales del software
Músicos	Profesionales que componen música y necesitan de herramientas para perfeccionar sus composiciones.	Uso directo del software
Usuario ordinario	Usuario que hace uso del software para entretenimiento o visualiza el resultado	Uso directo del software Uso indirecto del software
UDLA	Patrocinador de la idea de desarrollo de Sonido Visual	Uso indirecto del software

Resumen de usuarios

Nombre	Descripción	Afectado al que representa
Sordo	Hace uso del software como herramienta de ayuda en la percepción sonora.	Sordos
No sordo	Usuario del software, puede ser de forma directa o indirecta; como herramienta o para entretenimiento.	DJ's Músicos Usuario ordinario UDLA

Principales necesidades de los Afectados / Usuarios

Necesidad	Beneficio	Complejidad
“Escuchar” el sonido que se produce mediante ondas visuales.	Percibir como un todo el sonido, lo que permite interpretar el ritmo de la onda sonora.	Alta
Visualizar los canales sonoros que componen el sonido percibido.	Facilita la comprensión del timbre del sonido (agudos y bajos) y el volumen producido por la fuente sonora.	Alta
Personalizar elementos visuales.	El usuario determina el nivel del volumen y brillo del sonido traducido en elementos visuales.	media

Resumen del Producto

Perspectiva del producto

No definido

Supuestos y Dependencias

El entorno Quartz Composer incluye un parche para filtrar el sonido y entrega el espectro de sonido – con 12 canales – y el pico de volumen.

Licenciamiento e Instalación

La licencia de desarrollo de Quartz Composer es OpenSource, lo mismo aplica al entorno Xcode y el lenguaje de programación Objective-C.

La instalación se restringe a equipos Apple, ya que el entorno de desarrollo es perteneciente a la plataforma OS X.

Restricciones

- Equipos apple G4 (mínimo requerido)
- Sistema operativo OS X 10.4.6 (mínimo requerido)
- 512MB de memoría ram
- Tarjeta de video de 32MB (mínimo requerido, 64MB mínimo aconsejado)

Apéndice Uno – REQUERIMIENTOS QUE NO SE INCLUYEN (POSTPUESTOS)

Decidir el origen del sonido, entrada de audio o micrófono

Creación de un plug-in para el programa iTunes.

2 Descripción de Casos de Uso

Versión 2.0

Caso de Uso Visualizar Sonoridad

Breve Descripción

El caso de uso “Visualizar Sonoridad” define la funcionalidad del software: desplegar patrones gráficos y cromáticos como analogía a los sonidos captados.

Precondiciones

Primera precondición.

El sistema operativo debe ser OS X 10.4.6

Segunda precondición.

El equipo debe tener micrófono incorporado o entrada de línea de audio: si es así, debe tener un micrófono.

Postcondiciones de éxito

Primera postcondición.

El sonido es gráficado en la pantalla

Segunda postcondición.

El usuario puede modificar el volumen y ecualización de la gráfica.

Postcondiciones de falla

Primera postcondición.

La tarjeta gráfica no soporta el procesamiento

Segunda postcondición.

El usuario no comprendió el uso del software

Actor Principal

Usuario

Definiciones de acrónimos y abreviaturas

Ver el documento glosario de términos.

Requerimientos especiales

Utilización de recursos

El equipo debe constar con una tarjeta gráfica de 32MB (mínimo requerido) y 512MB Ram (mínimo aconsejado)

Diagramas

Caso de Uso Visualizar Sonoridad

Ver anexos, caso de uso

Diagrama de actividad Visualizar Patrones Sonoros

Ver anexos, diagrama de actividad

Pantalla

Ver anexos

Inclusiones

No existen

Extensiones

Controlar Ecualización: Permite al usuario modificar la ecualización gráfica

Controlar Volumen: Permite al usuario modificar la volumen gráfico.

Suposiciones y dudas pendientes

No existen.

Caso de Uso Controlar Ecualización

Breve Descripción

El caso de uso “Controlar Ecualización” define una funcionalidad extra del software: personalizar la ecualización gráfica de los elementos visuales de Sonido Visual.

Precondiciones

Primera precondición.

El programa Sonido Visual debe estar ejecutándose.

Segunda precondición.

El usuario debe comprender la iconografía para poder modificar la ecualización visual.

Postcondiciones de éxito

Primera postcondición.

La ecualización se modifica

Postcondiciones de falla

Primera postcondición.

El procesador no responde adecuadamente

Segunda postcondición.

El usuario no comprendió el uso del software

Actor Principal

Usuario

Definiciones de acrónimos y abreviaturas

Ver el documento glosario de términos.

Requerimientos especiales

Utilización de recursos

El equipo debe constar con una tarjeta gráfica de 32MB (mínimo requerido) y 512MB Ram (mínimo aconsejado)

Diagramas

Caso de Uso Visualizar Sonoridad

Ver anexos, caso de uso

Diagrama de actividad Visualizar Patrones Sonoros

Ver anexos, diagrama de actividad

Pantalla

Ver anexos

Inclusiones

No existen

Extensiones

No existen

Suposiciones y dudas pendientes

No existen.

Caso de Uso Controlar Volumen

Breve Descripción

El caso de uso “Controlar Volumen” define una funcionalidad extra del software: personalizar la volumen gráfico de los elementos visuales de Sonido Visual.

Precondiciones

Primera precondición.

El programa Sonido Visual debe estar ejecutándose.

Segunda precondición.

El usuario debe comprender la iconografía para poder modificar el volumen visual.

Postcondiciones de éxito

Primera poscondición.

El volumen se modifica

Postcondiciones de falla

Primera postcondición.

El procesador no responde adecuadamente

Segunda postcondición.

El usuario no comprendió el uso del software

Actor Principal

Usuario

Definiciones de acrónimos y abreviaturas

Ver el documento glosario de términos.

Requerimientos especiales

Utilización de recursos

El equipo debe constar con una tarjeta gráfica de 32MB (mínimo requerido) y 512MB Ram (mínimo aconsejado)

Diagramas

Caso de Uso Visualizar Sonoridad

Ver anexos, caso de uso

Diagrama de actividad Visualizar Patrones Sonoros

Ver anexos, diagrama de actividad

Pantalla

Ver anexos

Inclusiones

No existen

Extensiones

No existen

Suposiciones y dudas pendientes

No existen.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

Las conclusiones que se detallan no están clasificadas, corresponden a lo observado a lo largo del proyecto y por tanto buscan reflejar lo aprendido, así como lo obtenido.

- El conocimiento que se imparte y que se busca en el país está dirigido al desarrollo en plataforma Microsoft. Esto limita soluciones que puedan reducir costos a la inversión tanto de desarrollo como de investigación.

- El escaso conocimiento de varias plataformas por parte de muchos ingenieros o técnicos de sistemas por falta de investigación limita el desarrollo de aplicaciones de calidad convirtiendo a los profesionales de sistemas en asistentes de soporte para software que en muchos de los casos no es parametrizado a las necesidades del país.

- El temor al cambio por parte de los usuarios limita a los desarrolladores su campo de investigación y desarrollo.

- A pesar de ser el predecesor de Microsoft Windows, Apple continúa considerandose como una plataforma exclusiva para diseño.

- A pesar de ser un sistema operativo rediseñado sobre fundamentos *open source*, OS X sigue considerandose propietario, es decir la gente

cree que el sistema es “cerrado” y los desarrolladores consideran que no hay forma de acceder al sistema operativo y sus frameworks.

- El crecimiento que Apple ha tenido desde el año 2000 lo posiciona como una excelente opción para desarrollo de software.

- Cada vez el mercado de usuario Apple es mayor, lo que confirma la conclusión anterior.

- La conversión mental del sonido por colores (conocida como sinestecia) no logra ser comprendida por ciertas personas lo que limita la presencia de software como Sonido Visual.

- La gente sorda difícilmente comprende lo que es sonido y ni siquiera conoce lo que es ritmo o timbre, por tal motivo sin una adecuada instrucción de lo que significa cada elemento de un software de sinestesia, tal software no será aprovechado.

- A pesar de que a nivel internacional la educación musical para sordos es un elemento fundamental, acá en el país es considerado un mito el que un sordo pueda apreciar el sonido (al menos algunos elementos como ritmo, amplitud)

- El desarrollo de aplicaciones para OS X en el ambiente de desarrollo y aplicación – COCOA, es variado y extenso permitiendo crear software para distinto tipo de personas.

- La programación de un software de sinestecia, como lo es Sonido Visual, debe incluir en un futuro próximo ambiente 3D, es decir poder traducir el sonido a elementos visuales con espacialidad dentro del ambiente visual que presenta el monitor.

- El software de sinestesia puede ser una solución para ofrecer a las personas con deficiencias auditivas la oportunidad de percibir y disfrutar de los eventos musicales, así como del entorno sonoro en el que se desenvuelven, aunque no perciban ciertos elementos, el concebir el sonido como un elemento en movimiento es algo maravilloso para ellos.

- El proceso de enseñanza de un nuevo ambiente o nuevos conceptos a una persona sorda es más sencillo a mayor edad (17 – 25 años). Por tal motivo el manual debe incluir videos explicativos en lenguaje de señas que introduzca conceptos que una persona sorda desconoce, por ejemplo: ¿Qué es el sonido?

4.2 RECOMENDACIONES

- La Universidad de las Américas debe proveer conocimientos y facilidades para obtener nuevos criterios respecto al área de sistemas. Instalando en los mismos equipos PC particiones con sistema operativo Linux para prácticas e investigación de tal plataforma; del mismo modo instalar el paquete de desarrollo, como parte del DVD de instalación de OS X, en los equipos Apple, permitirá a los estudiantes conocer nuevas opciones de desarrollo profesional en el campo de los sistemas.

- Incentivar a los estudiantes a realizar aplicaciones distintas de las financieras o portales web, permitirá que los estudiantes expandan su visión comercial y creativa. Esto a largo plazo aumentará la cantidad, variedad y calidad del software realizado en el país.

- El país debe comprometerse con las necesidades de las personas discapacitadas promoviendo el desarrollo de aplicaciones que les permita involucrarse en actividades hasta el momento restringidas o incluso desconocidas por estos grupos minoritarios.

5 BIBLIOGRAFIA

Tanenbaum, Andrew S., “Sistemas Operativos Modernos”, ed. Prentice Hall, edición 1, 1992, pag. 1, pag. 2, pag. 300, pag. 301, pag. 302, pag. 303, pag. 306, pag. 308, pag. 356, pag. 357, pag. 358, pag. 361, pag. 362, pag. 441

García, Lidón, Peralta, Luis, Fernández, Samuel, “Un Paseo por la Historia” http://spisa.act.uji.es/~peralta/os/, 2000

wikipedia.org, “Silicon Valley”, http://en.wikipedia.org/wiki/Silicon_Valley, 2006

Tajnai, Carolyn E., “Fred Terman, The Father of Silicon Valley”, http://www.siliconvalley-usa.com/about/terman.html, 1995

Wikipedia.org, “Historia de los Sistemas Operativos” http://es.wikipedia.org/wiki/Historia_y_evolución_de_los_sistemas_operativos, 2006

TechNet, Microsoft, “Glossary” http://www.microsoft.com/technet/archive/wfw/7_agloss.mspx?mfr=true, 2006

Wikipedia.org, “Spooling”, http://es.wikipedia.org/wiki/Spooling, 2006

Wikipedia.org, “MS-DOS”, http://es.wikipedia.org/wiki/MS-DOS, 2006

Negus, Christopher, “Linux Bible”, editorial Willey Publishing, Inc., 2006, pag.40, pag. 42, pag. 43, pag. 46, pag. 49

Wikipedia.org, “History of Microsoft Windows”, http://es.wikipedia.org/wiki/history-of-microsoft-windows, 2006

Wikipedia.org, “History of Apple Inc.”, http://en.wikipedia.org/wiki/history-of-apple-inc, 2006

Wikipedia.org, “Mac OS history”, http://en.wikipedia.org/wiki/Mac_OS-History, 2006

Smigh, Amit, “A Brief History of Mac OS X”, http://www.kernelthread.com/mac/osx/history.html, 2006

Smigh, Amit, “XNU: The Kernel”, http://www.kernelthread.com/mac/osx/arch_xnu.html, 2006

Wikipedia.org, “History of Mac OS X”, http://en.wikipedia.org/wiki/history_of_Mac_OS_X, 2006

Singh, Amith, “Architecture of Mac OS X”, http://www.kernelthread.com/mac/osx/arch_xnu.html, 2006

Singh, Amith, “Architecture of Mac OS X”, http://www.kernelthread.com/mac/osx/arch.html, 2006

Apple Computer, “Introduction to Quartz Composer Programming Guide”, , Apple Reference Library, 2006

Quartz Compositions, “Introduction to Quartz Composer”, 2 http://www.quartzcompositions.com/phpBB2/mediawiki/index.php/Introduction_to_Quartz_Composer, 2005

Latour, Pierre-Oliver, “PixelShox Studio”, http://www.pol-online.net/pixelshox_techonology/index.php, 2003

http://developer.apple.com/documentation/GraphicsImaging/Conceptual/QuartzComposer/index.html?http://developer.apple.com/documentation/GraphicsImaging/Conceptual/QuartzComposer/qc_intro/chapter_1_section_1.html

Rees, Paul, Sparks, Fred, Rees, Charles, “Algebra”, editorial McGraw-Hill, ed. 1, 1991, pag. 149

Miyara, Federico, “Acústica y Sistemas de Sonido”, editorial Fundación Decibel, ed. 4, 2004, pag. 1, pag. 3, pag. 13, pag. 23, pag.24, pag. 27, pag. 28, pag. 64, pag. 68

Jacobson, Ivar, Booch, Grady, Rumbaugh, James, “El Proceso Unificado de Desarrollo de Software”, editorial Pearson Educación S.A., ed. 1, 2000, pag. 1, pag. 4, pag. 5, pag. 9, pag. 11

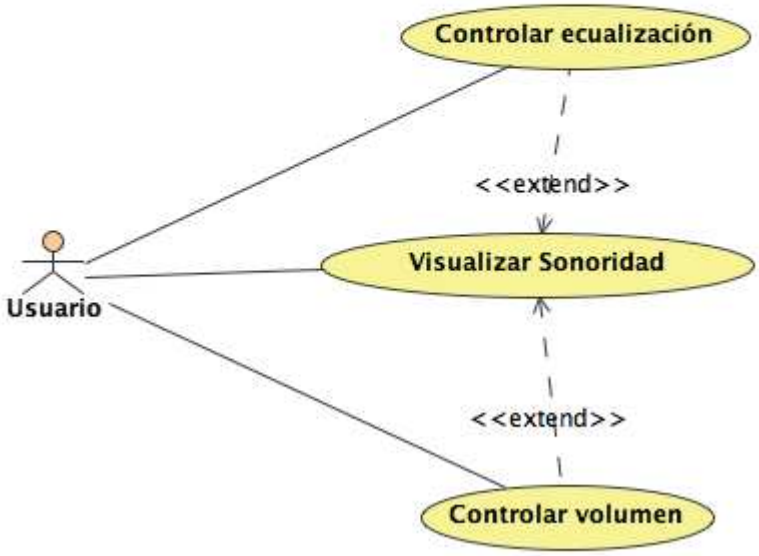
Apple Computer, “Mac OS X Technical Overview”, http://developer.apple.com/documentation/MacOSX/Conceptual/OSX_Technology_Overview/index.html?http://developer.apple.com/documentation/MacOSX/Conceptual/OSX_Technology_Overview/About/chapter_1_section_1.html, Apple Reference Library, 2006

Apple Computer, “Cocoa Fundamentals Guide”, http://developer.apple.com/documentation/Cocoa/Conceptual/CocoaFundamentals/Introduction/chapter_1_section_1.html, Apple Reference Library, 2006

6 ANEXOS

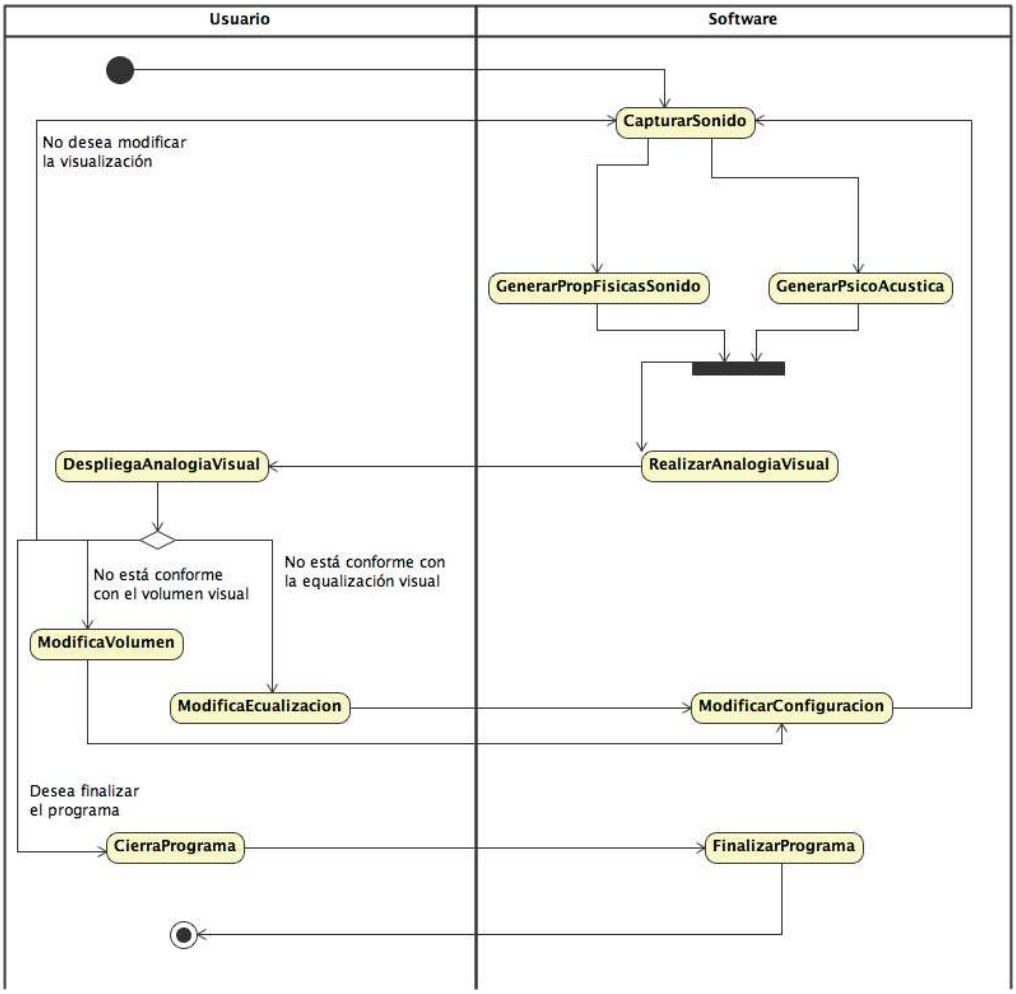
Anexo 1. Caso de uso Visualizar Sonoridad

Diagram name	Visualizar Sonoridad
Author	JaimeAndres
Creation date	8/21/07 9:45 PM
Modification date	9/7/07 10:19 PM

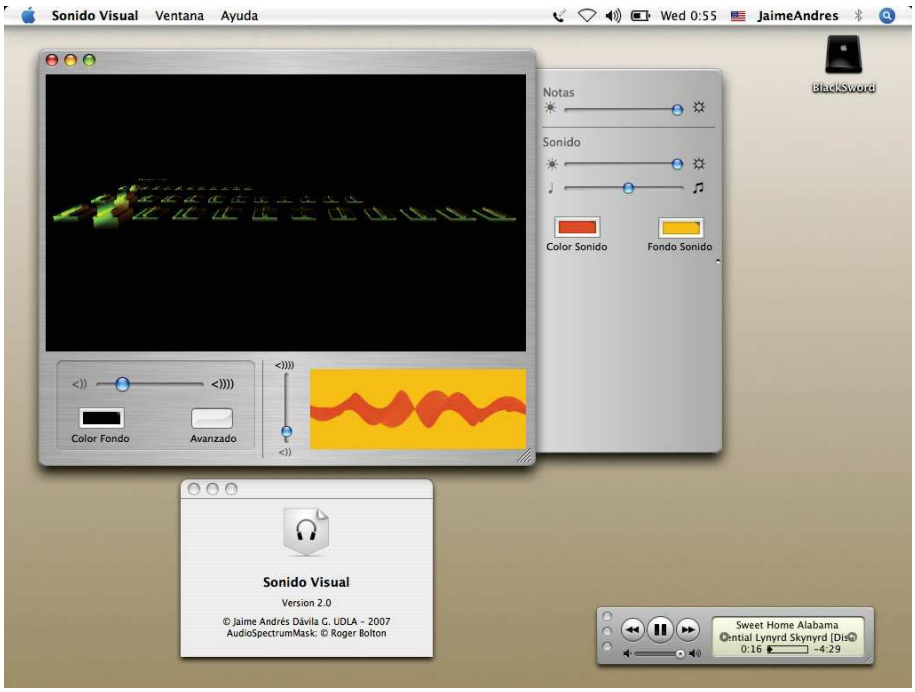


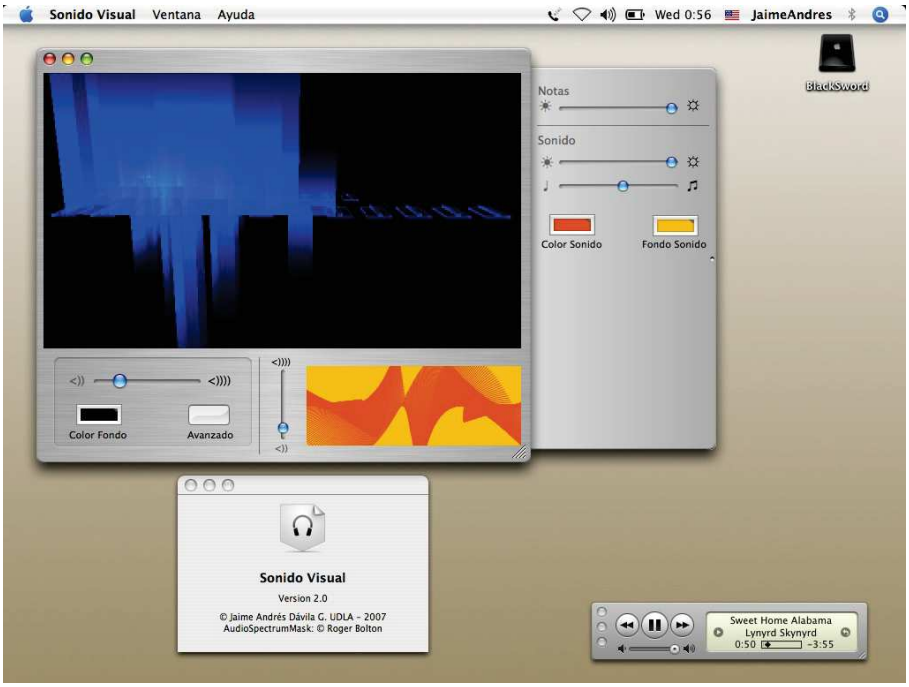
Anexo 2. Diagrama de Actividad Visualizar Patrones Sonoros

Diagram name	Visualizar Patrones Sonoros
Author	JaimeAndres
Creation date	8/21/07 11:12 PM
Modification date	9/7/07 10:18 PM



Anexo 3. Interfaz programa Sonido Visual





Anexo 4. Resultados obtenidos en niños sordos[105]

El día 7 de junio de 2007 se realizó una primera prueba con el primer prototipo funcional de Sonido Visual en el Instituto Nacional de Audición y Lenguaje, dirigido por la Dra. Rocío Torres. Se realizó la prueba con 8 jóvenes de sexto curso y la guía de la Dra. Rocío Torres, bajo las siguientes condiciones.

- Se reprodujo un video de Bond (un grupo de violinistas) mientras el programa traducía el sonido en formas visuales.
- Luego se configuró el equipo para que capte el sonido ambiental a través del microfono.

Estos fueron los resultados:

- En el instituto uno de los chicos indicó "…al hablar uno no se da cuenta que es posible captar el sonido a través de la vista… veo los sonidos producidos y los comprendo"
- "…los sonidos que produzco puedo verlos y se ven bonitos…"
- Al momento de ver el sonido ambiente descubrieron el ritmo y produjeron música.
- La profesora les enseño lo que es ritmo
- "…puedo ver como se van empatando lo que se ve (video musical) con las formas y colores producidos."

105 Para mayores detalles remitirse a la grabación que se encuentra en el CD del proyecto

El día 7 de septiembre de 2007 se realizó en el mismo instituto una segunda prueba con el producto en versión 2.0. En este caso los estudiantes fueron niños de primer curso. Cabe destacar que según la Dra. Rocío Torres, a esa edad los niños aún no han llegado a un nivel aceptable de comprensión del lenguaje por señas y esto se agrava ya que los no-oyentes no son "…usuarios del lenguaje español".

La prueba del software fue realizada de la siguiente forma:

- Se presentó diapositivas que explicaban una posible analogía entre los elementos sonoros – sonoridad, altura, timbre – con elementos visuales – color, saturación, brillo – y la forma en que tal analogía era plasmada por el programa.

- Como respuesta a la pregunta de comprensión, la gran mayoría indicó que el nivel de entendimiento fue como máximo 50%.

- Se reprodujeron dos videos de Bond y uno de Enrique Bumbury (se escogieron por las tomas visuales que se hacen a los instrumentos) y se solicitó que dibujasen lo que su imaginación haya percibido.[106]

- Se reprodujo la canción Sinfonía 40 de Mozart y se pidió que graficarán sus sensaciones mientras visualizaban el sonido.

106 Anexo 5

- Dentro del grupo se incluyó a un chico de 22 años, llamado José Luís Moreno quien, a pesar de haber llegado inesperadamente, disfrutó de la visualización e indicó: "…me parecía estar en un sueño…ver fuego que de pronto subía con fuerza y se convertía en nubes… nosotros como personas sordas nunca hemos percibido el sonido y solo miramos…"[107]

107 Para una revisión de la entrevista, revisar el CD con la pista entrevistaJoseLuis.mp3

Anexo 5. Gráficos de la percepción sonora