Arquitectura de una Interfase Usuario: Un Modelo General

Jorge A. Villalobos S.

Grupo de Investigación CGI/DAC
Departamento de Sistemas y Computación
Universidad de los Andes
Apartado Aéreo 4976 - Bogotá, COLOMBIA

Resumen: En este artículo se presenta un modelo general de una Interfase Usuario. El objetivo es definir un esquema da tal manera que cualquier tipo de interacción entre una aplicación y un usuario sea fácilmente expresable en términos del modelo. Esto implica que puede manejar cualquier tipo de dispositivos, y cualquier metodología de interacción (menus, comandos, etc.), lo mismo que eventos sincrónicos y asincrónicos, ayudas a todos los niveles, recuperación en caso de error, y todas aquella otras características deseables en una interfase usuario.

El presente escrito constituye la continuación de un trabajo desarrollado por el autor en el Institut National Polytechnique de Grenoble [VIL86].

Palabras Clave: Arquitectura de Interfases Usuario, Diseño de Interfases Usuario, Interacción hombre-máquina, Ingeniería de Software.

1. Introducción.

La interfase usuario es aquella parte de una aplicación que permite la interaccion entre el usuario y el resto del programa. El problema que existe actualmente es que debido a la masificación de la informática y a las grandes posibilidades hardware de interacción con el usuario, el diseño y la programación de la interfase están alcanzando costos considerables, llegando en algunos casos a constituir el 50% del código total de un programa.

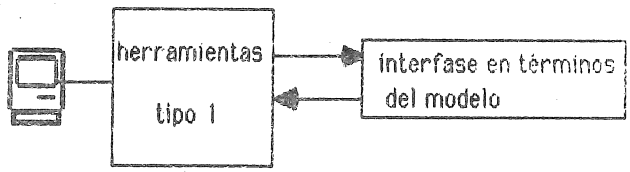
En el sistema tradicional de desarrollo de programas interactivos, el programador recibe una breve descripción de la interfase. A partir de ella, y utilizando lenguajes de programación que no cuentan a menudo con herramientas poderosas para este tipo de trabajo, debe implementar toda la interfase hasta niveles muy cercanos al nivel físico. Cada uno utiliza entonces su propio estilo y su propia interpretación de la especificación. Además, por la misma concepción de la aplicación, la interfase se encuentra repartida a lo largo de todo el programa haciendo imposible su adecuado mantenimiento y modificación. Esto conlleva en general muchos problemas sobre el producto final: dificultad en el mantenimiento de la interfase, imposibilidad de evaluar una especificación sin hacer su respectivo desarrollo, alto riesgo de error en la implementación, etc., además de las innumerables dificultades que existen para hacer una buena especificación de la interacción usuario-aplicación.

El primer paso que se ha dado para superar estos problemas ha sido el de utilizar métodos formales de especificación (gramáticas, BNF, autómatas), pero aun contando con una buena definición de la interfase el costo de su implementación sigue siendo alto y los problemas de mantenimiento quedan sin solución. Lo ideal en ese caso es poder ejecutar directamente la especificación o al menos poder generar automaticamente, a partir de ella, el código necesario para su ejecución.

En este artículo se pretende mostrar un modelo general de interfases usuario, basado en el modelo de niveles propuesto por Olsen [OLS83, OLS84] y Green [GRES5], y seleccionar un formalismo para poder representar cada nivel. También se incluye la definición de un esquema de comunicación entre ellos.

2. Para qué el modelo?

El objetivo final del proyecto que se está desarrollando, es la creación de un ambiente completo para la especificación y generación de interfases usuario. Para esto se ha decidido que la mejor manera de enfrentar el problema de diseñar las herramientas del ambiente de programación es crear como primera medida un modelo general de una interfase usuario y luego crear dos tipos de herramientas, las primeras que le permitan al programador (el usuario del sistema) expresar la interfase que desea desarrollar en términos del modelo. Si el modelo está bien construido, al igual que las herramientas del ambiente, el programador debe ser capaz de expresar fácilmente la interfase de cualquier sistema sin importar su tipo.



El segundo tipo de herramientas trabajan sobre cualquier interfase que se encuentre expresada en términos del modelo; le permiten al programador experimentar sobre ella (simular su ejecución, depurarla) y generar código en algún lenguaje de programación de alto nivel, que la ejecute.

El modelo debe entonces manejar de manera natural cualquier tipo de interacción entre el usuario y la aplicación, lo mismo que sistemas de ayuda dependiente del contexto (a varios niveles), recuperación en caso de error, cancelación de acciones etc

Resumiendo se puede decir que el modelo se hizo para definir las características y estructura del ambiente de programación de interfases, y en general para guiar la construcción de herramientas para la ayuda al diseño y creación de interfases usuario.

3. Características de una buena herramienta de diseño.

Las características que se consideran indispensables para cualquier herramienta de diseño de interfases (y en general para el diseño de software), y que nos deben por lo tanto guiar en la creación del modelo son las siguientes cuatro:

- El diseño y la especificación deben ser más fáciles de entender, crear y mantener que el programa que los ejecuta. La razon es que si es mas dificil diseñar la interfase que programarla directamente, no vale la pena utilizar las herramientas propuestas.

- Las herramientas de diseño deben ser suficientemente poderosas para poder especificar la interfase de cualquier sistema. Esto quiere decir que no se puede orientar hacia ningún tipo especifico de interacción (existen herramientas que solo trabajan sobre menus, por ejemplo), ni hacia ninguna clase particular de hardware

- Las herramientas deben permitir la ejecución directa de la especificación de la interfase o al menos la generación del código correspondiente. Sin esto pierde un poco de sentido el gastar tanto tiempo en la tarea de crear una buena especificación, puesto que los problemas se van a presentar luego en las fases de implementación y mantenimiento.

- El modelo en el que se basan las herramientas debe estar fuertemente ligado al modelo mental que el programador asocia a la interfase, o más exactamente al proceso de su diseño. Esto implica que se descartan de plano las gramáticas y las BNF propuestas en algunos sistemas, dada su dificultad de utilización para especificar una interfase.

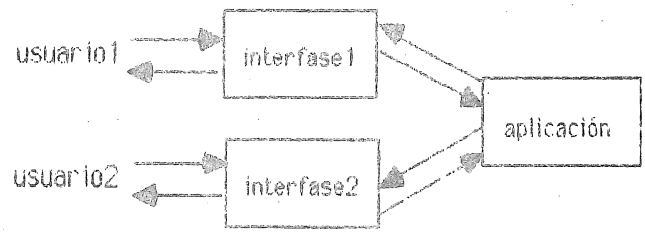
Teniendo en cuenta estas características y otras consideraciones como portabilidad, modularidad, etc que exige la ingeniería de software, se propone el siguiente modelo para una interfase usuario.

4. Estructura general del modelo.

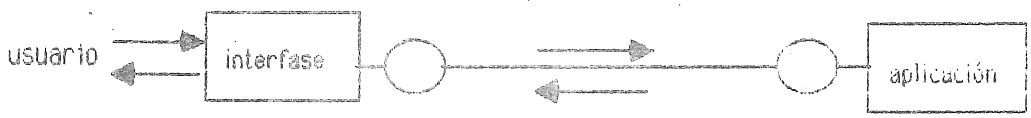
De acuerdo con la definición dada, la interfase es la parte del sistema encargada de la interacción entre la aplicación y el usuario.

Para poder garantizar independencia y modularidad de la interfase, vamos a exigir una separación clara entre ésta y la aplicación, encontrándose relacionadas únicamente mediante un esquema formal de comunicación.

Esto quiere decir que una misma aplicación puede ser manejada por dos interfases distintas (dependiendo posiblemente del tipo de usuario) sin necesidad de ser modificada.



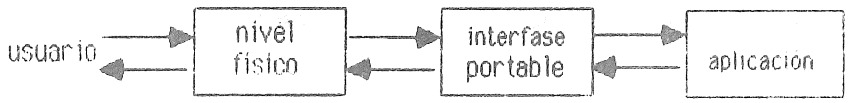
llegando incluso en un sistema distribuido a permitir que la interfase se encuentre en el nodo del usuario y la aplicación en un nodo diferente de la red,



sin necesidad de variar ni la aplicación ni la interfase, sino únicamente adoptando un nuevo esquema de comunicación entre los dos.

Además de éso es necesario, para poder garantizar independencia del hardware (portabilidad), separar de la interfase todos aquellos detalles dependientes del nivel

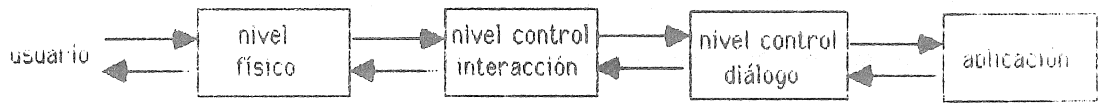
físico de tal manera que para llevarlo a otro equipo solo sea necesario redefinir esta parte. Para facilitar ésto, la interfase debe contar con una descripción completa del hardware de que dispone, librerías de rutinas que permitan la adecuada manipulación de los dispositivos y referirse en el resto de la interfase a los dispositivos en términos lógicos y nunca físicos. Esta parte se denomina el nivel físico de la interfase



A su vez hay dos partes claramente distinguibles dentro de la interfase cuando ya no depende de los detalles hardware. La primera se denomina el control de diálogo que indica la manera como debe ir moviendose la interfase del sistema segun las entradas del usuario y las posibilidades de la aplicación. Si se ve esta última simplemente como un conjunto de rutinas, la misión de este nivel es el de "oir" las ordenes del usuario, y decidir que procedimientos de la aplicación se deben llamar y en que orden, para satisfacerlo.

Por último, existe otro nivel dentro de la interfase que es el encargado de interactuar directamente con el usuario. Debe oirlo e interpretarlo, informando luego al nivel de control de diálogo lo que el usuario quiere. Este nivel se denomina el de control de interacción.

Estos dos últimos niveles son independientes entre si, en el sentido de que al control de diálogo no le interesan los detalles de cada interacción sino solo los resultados (no le interesa si el usuario escoge una opción de un menú, o si da un comando completo, o si adoptó la respuesta por defecto, o por cual dispositivo dió la respuesta), mientras que al nivel de control de interacción no le importa para que es utilizada la respuesta o la manera como será ejecutada por la aplicación.



En la bibliografía del tema, la aplicación se denomina el nivel semántico de la interfase, el nivel de control de diálogo se llama el nivel sintáctico y el nivel de control de interacción el nivel léxico. Estos nombres se encuentran inspirados en las

fases de análisis de los compiladores, y en la idea de algunos autores acerca de que una interfase no es más que un compilador de un lenguaje dinámico mediante el cual se comunica un usuario con una aplicación.

Lo importante en el modelo es la independencia entre sus diferentes niveles. Esto permite, en el momento de buscar un formalismo para expresar cada uno, escoger el más adecuado para cada tarea sin tener que pensar en una buena manera de expresar todas las características de una interfase al mismo tiempo.

Una vez definida la estructura general de una interfase se deben resolver dos problemas: el primero es decidir la forma de comunicar los niveles y el segundo es diseñar un formalismo que se ajuste a las necesidades y objetivos de cada nivel, para especificar cada una de las partes que componen la interfase. En un compilador, por ejemplo, se utilizan BNF para la parte sintáctica y expresiones regulares para la parte lexicográfica.

5. El nivel físico.

Este nivel es el encargado de garantizar la independencia entre la interfase y el hardware. Mantiene una descripción de cada uno de los dispositivos que maneja el sistema, a cada uno de los cuales le asocia un nombre mediante el cual es reconocido en el resto de la interfase. Cuenta ademas con un conjunto de rutinas que le permiten manejar cualquiera de los dispositivos presentes (dada su descripción).

Los dispositivos se dividen en dos: dispositivos de entrada y dispositivos de salida. Para cada dispositivo de entrada se debe especificar:

- nombre
- tipo: señala a que clase de dispositivo pertenece, entre las cinco posibilidades que define GKS (locator, valuator, choice, string, pick)
- rutinas de manejo del dispositivo
- características hardware especiales

Por su parte los dispositivos de salida se definen mediante los siguientes tres componentes:

- nombre
- rutinas de manejo del dispositivo
- características hardware

La comunicación hacia el nivel de control de interacción es de la forma:

(dispositivo, contenido)

que indica el nombre del dispositivo que recibió la entrada y el contenido de la misma.

No es el objetivo de este nivel interpretar la entrada del usuario, sino simplemente recibirla por uno de sus dispositivos y transmitirla al nivel superior

6. El nivel de control de comunicación.

Este nivel es el encargado de manejar la comunicación entre el usuario y el control de diálogo. Para ésto recibe una señal de activación desde el nivel superior, se comunica con el usuario para obtener una respuesta, la interpreta y luego la manda al control de diálogo.

El nivel se encuentra conformado por un conjunto de objetos, cada uno de los cuales sabe manejar una interacción con el usuario. El que decide cuando y cual de todos los objetos se activa en un momento dado es el nivel superior.

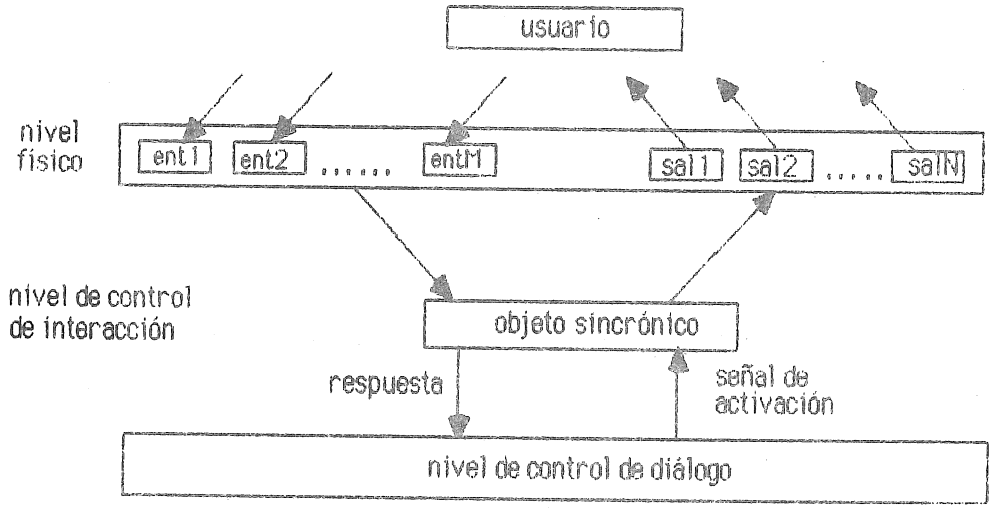
Los objetos son entonces independientes entre ellos y autosuficientes en el sentido de que saben cómo interpretar una entrada, cómo recuperarse de los errores, cómo informar al usuario que están activos, etc., descargando de todos estos detalles al control de diálogo.

Existen dos tipos de objetos en el nivel: los objetos sincrónicos y los asincrónicos. Los primeros cuando se activan toman temporalmente el control de la interfase y solo lo retornan cuando tienen una respuesta del usuario o un informe de fracaso en la interacción. Los asincrónicos cuando se activan retornan inmediatamente el control

al nivel superior y quedan dormidos hasta cuando reciben una entrada especifica del usuario. En ese momento interrumpen el control de diálogo informándole la señal dada por el usuario.

6.1. Objetos sincrónicos

Un objeto sincrónico está definido por dos partes: la interfase con los dos niveles que lo rodean y su comportamiento interno. Gráficamente la ubicación de un objeto sincrónico dentro de la estructura de la interfase sería la siguiente:



Cuando llega la señal de activación, el objeto selecciona uno o varios de los dispositivos de salida asociados e informa al usuario que se encuentra activo (presenta un menú, o le hace una pregunta, etc.). Luego coloca los dispositivos de entrada asociados al objeto en estado de alerta y por el primero que llegue una respuesta la toma, la interpreta y luego la envía al nivel de control de diálogo.

La comunicación con el nivel físico tiene entonces la siguiente forma:

(disp.salida1,contenido1, ... , disp.salidaN,contenidoN, disp.entrada1, ... , disp.entradaM)

El nivel físico sabrá, partiendo de la descripción de los dispositivos y utilizando las rutinas asociadas a cada uno, cómo manejar la comunicación directa con el usuario.

La comunicación hacia el nivel sintáctico tiene la siguiente forma:

(código de retorno, respuesta)

donde incluye la respuesta del usuario ya interpretada (ya no importa el dispositivo por el que llegó, ni la entrada física dada por el usuario, sino solamente lo que él quiere decir) y un indicador del éxito o del tipo de problema que se presentó durante la interacción con el usuario.

El comportamiento interno de un objeto es la segunda parte que se debe considerar. Dicho comportamiento es el que define la manera de reaccionar del objeto ante las diferentes situaciones que se pueden dar. Se encuentra compuesto por un conjunto de métodos que incluyen la manera de interpretar la entrada (dado el dispositivo que la recibió) y la forma de recuperarse en caso de error. Si este último no se encuentra definido en un objeto, al detectar un error simplemente lo comunica como parte del código de retorno al nivel superior. La recuperación de error puede consistir en un reintento de interacción con el usuario, la construcción de una respuesta por defecto o la misma cancelación de la ejecución de la interfase.

Existen algunas características de un objeto que solo pueden ser definidas durante la ejecución misma de la interfase. Si se trabaja por ejemplo con un objeto que presenta un menú con un número variable de opciones, dicho valor solo se conocerá en el momento de activar el objeto. Estas se denominan sus características dinámicas, y deben ir definidas como parte de la señal de activación que envía el nivel superior. Esta señal tiene por lo tanto la siguiente forma:

(nombre del objeto, características dinámicas, tiempo máximo de espera)

en la cual se incluye también un tiempo límite que se le dá al usuario para dar una entrada (timeout), y después del cual los métodos internos de comportamiento deciden como reaccionar.

6.2. Objetos asincrónicos

Un objeto asincrónico tiene una estructura un poco diferente a la expuesta anteriormente, dado que su misión no es comunicar información al nivel superior, sino control.

Existen varias tipos de señal que puede mandar el control de diálogo a un objeto asincrónico. La primera es:

(ACTIVAR, nombre del objeto, forma de interrumpir, prioridad)

mediante la cual se indica cual objeto se activa, como debe reaccionar al recibir la señal del usuario (puede dar control a una rutina del nivel semántico, o pasar a algun punto específico del control de diálogo (ver estados seguros), etc.) y cuál es su prioridad con respecto a los demás objetos asincrónicos activos. De otra parte la señal.

(DESACTIVAR, nombre del objeto)

desactiva el correspondiente objeto asincrónico.

La definición de un objeto asincrónico consta únicamente de un posible mensaje de activación hacia el usuario (por uno o varios dispositivos de salida) y de la definición de por dónde y mediante cuál entrada se debe activar.

7. El nivel de control de diálogo

El nivel sintáctico o de control de diálogo tiene como misión relacionar las entradas del usuario con las rutinas semánticas (de la aplicación), guiando de esta manera toda la ejecución de la interfase. En este nivel, dado un contexto de ejecución y una entrada del usuario, se decide que rutinas semánticas invocar y que nuevo estado alcanzar.

A este nivel de la interfase es al cual le han dedicado más trabajo en la bibliografía que existe del tema, y casi siempre cuando se habla de interfase usuario se refieren específicamente a este nivel. Como formalismo para especificarlo utilizan BNF,

gramáticas y autómatas. En este artículo se va a presentar un formalismo de diagramas de transición aumentados, con un poder computacional equivalente al de una máquina de Turing. Esto implica que cualquier interfase que se pueda expresar en un lenguaje de programación, se puede definir en términos del modelo.

El nivel completo está compuesto por un conjunto de diagramas de transición y por un espacio de memoria compartido por todos. Uno de los diagramas de transición se conoce como el inicial, y es por éste que comienza siempre la ejecución de la interfase. Incluye además un primer nivel de ayuda que corresponde a otro de los diagramas de transición del nivel, y su misión es controlar la presentación de la ayuda más general del sistema.

Un diagrama de transición por su parte está compuesto por 7 elementos:

- nombre: forma de identificarlo desde los demás diagramas
- segundo nivel de ayuda: nombre del diagrama que contiene el segundo nivel de ayuda
- conjunto de estados (E)
- conjunto de estados seguros (S, $E \supseteq S$)
- estado inicial (I, $I \in E$)
- conjunto de estados de retorno (R, $E \supseteq R$)
- conjunto de transiciones entre los estados (T)

Un estado de un diagrama de transición representa un estado posible de ejecución de la interfase. Consta de los siguientes elementos

- nombre: forma de identificarlo desde los demás estados
- precondición: aserción que se debe cumplir al llegar al estado
- acciones sobre el estado: corresponden a llamadas de rutinas semánticas, llamadas a diagramas de transición del nivel o señales de activación a los objetos del nivel de control de interacción.
- tercer nivel de ayuda: nombre de un diagrama de transición que maneja una ayuda específica del estado
- postcondición: aserción que se debe cumplir al abandonar el estado.

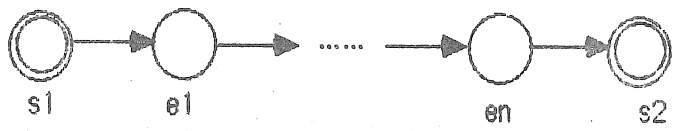
Si al llegar a un estado en un diagrama de transición, no se cumple la precondición, o si al salir no es verdad lo que se exige en la postcondición, la interfase cancela su ejecución.

Por su parte una transición indica un cambio en el estado de ejecución de la interfase causado por alguna entrada del usuario o por alguna otra situación interna. Una transición se encuentra definida por:

- prioridad: en caso de que existan dos posibles transiciones activables en un contexto dado, se toma la de mayor prioridad.
- estado de salida
- estado de llegada: estado al cual llega por la transición
- guarda de la transición: condición que debe satisfacerse para poder activar la transición.
- acción sobre la transición: corresponde a la llamada de una rutina semántica o de un diagrama de transición, que se requiere para poder cumplir la precondición del estado que se va a alcanzar.

Si en un momento dado de la ejecución no se encuentra ninguna transición activable, el control de diálogo suspende la ejecución completa de la interfase.

En cada diagrama de transición puede existir un conjunto de estados seguros, que se comportan de la siguiente manera: todo lo que se haga en el contexto entre dos estados seguros solo se confirma al llegar al segundo. Esto permite el manejo de operaciones de manera atómica (todo o nada), puesto que desde cualquier estado intermedio es posible regresar a la situación que se tenía en el último estado seguro.



Si como parte de las rutinas semánticas existen manejadores de Bases de Datos, por ejemplo, estos solo serían llamados en los estados seguros mientras en los estados intermedios se utilizaría algún tipo de mecanismo (manejar una copia, guardar todo en un archivo intermedio, etc.) para almacenar las transacciones parciales Por

efectos prácticos, todo estado de retorno se considera seguro.

Al alcanzar un diagrama de transición un estado de retorno, termina su ejecución y retorna el control al punto donde fue llamado. En todo diagrama debe existir por lo menos un estado de retorno.

Mediante la utilización de los objetos asincrónicos es posible manejar tanto los estados seguros como la ayuda del sistema, la cual tiene tres niveles: ayuda general a nivel de todo el sistema, ayuda a nivel del diagrama de transición y ayuda particular del estado en el que se encuentra.

8. El nivel semántico

El nivel semántico está compuesto por el conjunto de rutinas de la aplicación. Estas son activadas desde el nivel sintáctico el cual envía además información adicional de su memoria compartida como parámetros a las rutinas:

(nombre de la rutina, parámetros)

Dependiendo de la definición del procedimiento, algunos de estos parámetros pueden ir por referencia y por lo tanto ser modificados dentro de la rutina. No existe ningún otro tipo de comunicación entre los niveles semántico y de control de diálogo.

9. Conclusiones

El modelo que se presenta en este artículo es suficientemente general como para abarcar cualquier tipo de interfase. Existen muchos puntos que no se concretan debido a limitaciones de espacio. Se está esperando la terminación del ambiente de diseño y generación de interfases basado en el modelo, que se encuentra en desarrollo dentro del Grupo de Investigación CGI/DAC de la Universidad de los Andes, para hacer la respectiva evaluación y posible realimentación.

10. Bibliografía

[OLS83] Olsen, D., Dempsey, E., Syntax Directed Graphical Interaction, Proceedings of the SIGPLAN'83, Symposium on Programming Language Issues in Software Systems, Sigplan Not. 18, 6, pp. 112-117, Junio/1983.

[OLS84] Olsen, D., Pushdown Automata for User Interface Management, ACM Transactions on Graphics, Vol. 3, num. 3, pp. 177-203, Julio/1984.

[VIL86] Villalobos, J., Automate pour Application Interactive, Rapport de D.E.A., Institut National Polytechnique de Grenoble, Julio/1986.

[GRE85] Green, P., The University of Alberta User Interface Management System, SIGGRAPH'85, Vol. 19, num. 3, pp. 205-213, Julio/1985.

[BUX83] Buxton, W., Lamb, M., Towards a Comprehensive User Interface Management System, SIGGRAPH'85, Computer Graphics, Vol. 17, num. 3, pp. 35-42, Julio/1983.

[COU84] Coutaz, J., A Paradigm for User Interface Architecture, Department of Computer Science, Carnegie-Mellon University, Mayo/1984.