UNIVERSIDADE DE VIGO

Departamento de Informática
Área de Lenguajes y Sistemas Informáticos

REPRESENTACIÓN Y ANÁLISIS DE LA COMPONENTE VISUAL DE LA INTERFAZ DE USUARIO

TESIS DOCTORAL

Javier Rodeiro Iglesias

Director: Manuel Pérez Cota

Septiembre de 2001

A Tony e Iván

Os meus devanceiros que fixeron posible que chegase aqui.

A miña familia (Tony moito levou, pero a Iván tamén lle chegou), por aturarme durante meses nos que levaba os problemas do traballo para a casa.

O R.C.M. San Bartolomé y Santiago de Granada e especialmente ó seu director D. José Luis Pérez Serrabona polo agarimo e coidado que da miña persoa tivo no tempo no que abafaba de calor por ala.

Á Asociación de Interacción Persona Ordenador e á Sección Española de Eurographics polo apoio que me diron durante a realización da tese.

Ás áreas de Linguaxes e Sistemas Informáticos das Universidades de Granada e Castellón polos seus comentarios e suxerencias durante o traballo de investigación.

Ó Director e ós meus compañeiros do Departamento de Informática da Universidade de Vigo, e dentro deles o director e o secretario da miña tese, xa que so se mollan aqueles que se meten no rio.

A Eva por todalas catro y cinco da maña de traballo e por non cansarse nunca de revisar, criticar, cuestionar, evaluar, discutir, animar, apoiar, escribir, aturar e berrar conmigo

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Objetivos y justificación

1.1 Presentación

Existen una gran cantidad de técnicas de representación de la interfaz de usuario de un sistema interactivo. Estas técnicas, muchas de ellas dentro de marcos formales, abarcan la especificación y validación de las distintas fases a seguir dentro del proceso de creación de interfaces de usuario gráficas, englobando desde el estudio del usuario a nivel de modelos mentales hasta la representación de su visualización final en los dispositivos de visualización.

Las técnicas actuales contemplan varias fases dentro del proceso de diseño, pero hasta el momento ninguna todas. En general estas técnicas abarcan desde la primera fase, el estudio del usuario, hasta la fase de diseño lógico de los componentes de la interfaz, o bien desde la fase de diálogo hasta la representación concreta en el dispositivo de visualización. Algunas técnicas inciden únicamente en las fases intermedias del desarrollo de la interfaz de usuario y dejan el estudio inicial y la implementación para otros expertos. Prácticamente todas incorporan mecanismos de testeo y validación de su representación y algunas generan documentación de gran utilidad y aportan herramientas que las hacen prácticas.

Sin embargo, no existen muchos estudios acerca de la validez y corrección de la representación concreta de la interfaz sobre los dispositivos de visualización, ni existen mecanismos de representación abstracta de los componentes individuales de la interfaz de usuario ni de su composición para dar lugar a una interfaz final que será utilizada

por el usuario.

En la actualidad, para lograr la definición de una interfaz de usuario (ya sea abstracta o final) existen dos métodos. Por un lado, está la utilización de herramientas que definan la apariencia visual y el comportamiento de los componentes de la interfaz. Dichas herramientas utilizan, en su mayoría, funciones de prototipado para la construcción de la interfaz y realizan la visualización de los componentes directamente sobre una plataforma. Además, determinan su colocación dentro de la interfaz de forma concreta, aunque provean de capacidad al usuario para su manipulación. Una vez generada la apariencia visual sobre una plataforma, su colocación sobre el dispositivo y los eventos a los cuales responderán, se consigue una interfaz de usuario final completa.

Otra opción es utilizar modelos de representación abstracta de la interfaz que abarcan varias de las fases en el diseño de interfaces de usuario como se ha mencionado anteriormente. En la mayoría de los casos únicamente indican la existencia de una función de visualización para el componente que modelan; en otros casos determinan de forma abstracta aspectos de su apariencia visual, pero la colocación de los componentes definidos de forma individual no tiene en cuenta las posibles manipulaciones que el usuario pueda realizar sobre los mismos y que no correspondan a la interfaz.

En cualquiera de los dos casos el objetivo de lograr una definición única e independiente de la plataforma de desarrollo y el dispositivo de visualización no es posible. Una definición única implica que una vez comunicada a dos implementadores la interfaz final debe ser igual. Una definición independiente implica que pueda ser portable e implementable en cualquier sistema.

La práctica totalidad de los sistemas de representación de interfaces consideran sistemas gráficos que son estáticos en su visualización. Sin embargo, los sistemas de visualización y las aplicaciones actuales permiten al usuario la manipulación de la componente visual de aquello con lo que están interactúando. Por ello deben proporcionarse elementos de representación de la composición (entendido en el sentido de colocación) y que soporten la manipulación por parte del usuario. Entre estos elementos se distinguen la representación de la apariencia, la composición y el diálogo de los componentes de la interfaz. También debe existir un grafismo de la interrelación entre los componentes para poder observar y estudiar desde una perspectiva global la interfaz de usuario.

1.2 Objetivos

El objetivo del presente trabajo es desarrollar un modelo de representación de un sistema interactivo. Existen una gran cantidad de variables visuales que intervienen en el proceso de percepción visual y, por lo tanto, en la capacidad de obtención de información, su procesado y la respuesta gestual a la misma. Una interfaz que tenga una componente visual difícil de entender por parte del usuario puede hacer que el producto tenga un éxito escaso, independientemente de su calidad y utilidad. Por otra parte, si la componente visual no permite una interacción clara con el usuario, este encontrará el sistema complejo y el esfuerzo suplementario necesario para poder realizar las tareas puede hacer que el usuario no utilice finalmente dicho sistema.

El trabajo aquí presentado cubre la definición abstracta de la apariencia visual que es tratada superficialmente en algunos modelos actuales. Esta definición de la apariencia se realiza a través de la definición de su geometría y los atributos visuales de la misma. Proporciona un sistema de colocación de componentes jerárquico y abstracto y una representación del diálogo enfocado desde la percepción visual de la interfaz por parte del usuario.

Una representación abstracta de los componentes y de su composición tiene importantes ventajas. Por un lado, permite un estudio previo a la implementación acerca de la composición de la interfaz de usuario desde una perspectiva estética y de usabilidad. Por otro lado, una representación abstracta nos brinda la posibilidad de desarrollar herramientas que utilicen esta representación para analizar aspectos funcionales y estéticos de la interfaz y para generar los componentes de forma automática. La representación del diálogo permite identificar las tareas y aspectos relativos a la visualización final como la jerarquía de dependencia y visual de la interfaz. La jerarquía de dependencia representa gráficamente la relación de inclusión que existe entre los componentes de la interfaz, y la jerarquía visual representa la existencia de capas visuales en la interfaz y la interrelación entre las mismas.

El trabajo realizado no está condicionado por ningún soporte hardware ni sistema software pasado ni actual. Uno de los principales problemas encontrados es abstraer observaciones y premisas de sistemas de amplia difusión y que son considerados de uso estándar. Tampoco se pretende asociar esta representación con herramientas particulares sea cual sea su finalidad (herramientas de autor, herramientas visuales de desarrollo, sistemas operativos o editores gráficos) sino todo lo contrario.

Con el presente trabajo se analiza la componente visual del sistema, de forma que se puedan identificar las relaciones entre los componentes visuales individuales y los atributos que poseen. Con la identificación de las relaciones y los atributos podremos indicar la adecuación de dichos componentes visuales a los principios que determinan una correcta construcción de las interfaces visuales, y dar una evaluación cuantitativa de la corrección de esa interfaz a nivel visual estético.

1.3 Estructura de la tesis

Esta tesis se divide en las siguientes partes:

- En el capítulo dos se expone una definición de sistema gráfico interactivo, con una descripción de cada uno de los participantes del sistema interactivo y el papel que juegan en el mismo en relación con la importancia que esto tiene en el desarrollo de interfaces de usuario.
- A continuación, en el capítulo tres se realiza un estudio sobre los actuales modelos y herramientas de representación de sistemas interactivos. Se realiza una descripción de fases cubiertas por cada uno de ellos y se incide sobre todo en la identificación de sistemas que representan y realizan una evaluación de la componente visual de las interfaces de usuario.
- En el capítulo cuatro se define un sistema de representación de la interfaz de usuario a nivel de la componente visual del mismo que incluye el diálogo producido, a través de eventos, entre la misma y el usuario.
- En el capítulo cinco se establecen una serie de parámetros de estudio de la interfaz de usuario sobre la representación definida en el capítulo anterior con el objeto de demostrar su aplicabilidad para la realización de análisis de la interfaz. Entre los análisis establecidos se encuentran la evaluación de parámetros visuales, la cardinalidad de componentes, la jerarquía de capas visuales, la identificación de estados de la interfaz y de las tareas posibles.
- En el capítulo seis se establecen las conclusiones obtenidas después de la realización de este trabajo y se definen las líneas futuras de trabajo que se abren a partir del mismo.

Capítulo 2

Sistemas visuales interactivos

2.1 Interacción Persona-Ordenador

Cuando los seres humanos y los ordenadores interactúan lo hacen a través de un medio o interfaz. Una interfaz es una superficie de contacto [Lau90] que refleja las propiedades físicas de los que interactúan, las funciones a realizar y el balance de poder y control.

En el caso de la Interacción Persona-Ordenador, la interfaz es el punto en el que seres humanos y ordenadores se ponen en contacto, transmitiéndose mutuamente tanto información, órdenes y datos como sensaciones, intuiciones y nuevas formas de ver las cosas. Autores como Moran [MC96] la definen como *"los aspectos del sistema con los que el usuario entra en contacto"* , Chi [Chi85] dice que es *ün lenguaje de entrada para el usuario, un lenguaje de salida para el ordenador y un protocolo para la interacción"* y Negroponte [Neg94] en su libro "Being digital"indica *"La interfaz es el sitio donde los bits y las personas se encuentran"*.

La disciplina de Interacción Persona-Ordenador (IPO) se conoce en la comunidad internacional como Human-Computer Interaction (HCI) o Computer-Human Interaction (CHI). Para el mundo hispanohablante se ha adoptado la expresión de Interacción Persona-Ordenador y como acrónimo IPO. El razonamiento que se ha seguido es el siguiente [AAC+01]: *Human* podría ser traducido como *Hombre*, pero tiene connotaciones sexistas. Por otro lado, *Ser humano* es demasiado largo así que se ha optado por *Persona*, término que el diccionario define como individuo de la especie humana.

En cuanto a la segunda parte, *Computer*, aunque en ocasiones ha sido traducido por máquina, este término es demasiado amplio y por tanto se considera más adecuado computador u ordenador. De entre ambos parece que ordenador es un término que se ha impuesto con más claridad en España y por tanto se llega a la expresión de Interacción Persona-Ordenador (IPO).

La ACM (Association for Computer Machinery) es, posiblemente, la organización internacional de investigadores y profesionales interesados en todos los aspectos de la computación más importante del mundo. Esta asociación tiene un grupo especial de trabajo en temas de IPO denominado SIGCHI (Special Interest Group in Computer Human Interaction) el cual propuso la siguiente definición de Interacción Persona-Ordenador: *"Es la disciplina relacionada con el diseño, evaluación e implementación de sistemas informáticos interactivos para el uso de seres humanos, y con el estudio de los fenómenos más importantes con los que está relacionado"*.

En España, la asociación más importante que aglutina a los investigadores del ámbito de la IPO es la Asociación de Interacción Persona-Ordenador (AIPO). Esta asociación desarrolla, entre otras actividades, la organización de congresos internacionales, coordina grupos de investigación en IPO y ha desarrollado un curso de Introducción a la Interacción Persona-Ordenador [AAC+01] que sirve de guía para la docencia de la IPO dentro de la comunidad universitaria de habla hispana (muchos de los conceptos aquí abordados están reflejados y han sido obtenidos de este curso).

De entre todos los conceptos relacionados con la IPO se distinguen dos que tienen especial relación con el presente trabajo: el diseño y la consistencia. El Diseño es una actividad encaminada a conseguir la producción en serie de objetos útiles y bellos. Tal como se entiende actualmente, pretende actuar sobre el entorno físico del hombre para mejorarlo en su conjunto. La Interfaz de Usuario (IU) es la puerta del usuario a la funcionalidad del sistema subyacente. Las interfaces de usuario mal diseñadas son un factor que frena el uso de las funcionalidades. La Consistencia es un concepto clave en la usabilidad de un sistema informático. Diremos que un sistema es consistente si todos los mecanismos que se utilicen en un sistema sean siempre usados de la misma manera. Recomendaciones para diseñar sistemas consistentes:

- Seguir guías de estilo siempre que sea posible.
- Diseñar con un "look & feel"(apariencia y comportamiento) común.

- No hacer modificaciones sobre nuevas versiones si no es necesario hacerlas.
- Añadir nuevas técnicas al conjunto preexistente, en vez de pedir al usuario que cambie las que conoce.

2.2 Los sistemas interactivos

Un sistema interactivo es aquel que soporta la comunicación entre el computador y el usuario, en ambas direcciones, permitiendo seguir los pasos en la actividad del usuario. Cuando éste realiza acciones, como pulsar una tecla, el sistema reacciona de acuerdo con esas acciones, escribiendo un carácter en pantalla por ejemplo. Todo esto tiene lugar en la interfaz de usuario, la parte del sistema que da acceso a los recursos internos de la máquina.

La interfaz puede utilizar diferentes medios para comunicarse con el usuario. Si envía información a través del dispositivo de visualización el usuario la recibe a través del sentido de la vista. En cambio, si la información enviada es sonido (tal como un mensaje pregrabado de error) el usuario utiliza el sentido del oído. Cuando el usuario produce una acción sobre el sistema puede hacerlo mediante la selección sobre un elemento visual, a través de un dispositivo apuntador, o bien a través de teclas. Dependiendo del sistema perceptivo y sistema motor que el usuario utilice, para recibir información del sistema o realizar acciones sobre la interfaz, se produce interacción sobre una parte u otra de la interfaz de usuario.

Aunque la definición inicial se centra sobre un computador, la interfaz de usuario puede corresponder también a cualquier máquina que utilice un procesador para gestionar su funcionamiento e incorpore mecanismos de comunicación con el usuario. Ejemplos de estas computadores de propósito genérico son fotocopiadoras, cajeros automáticos, microondas, hornos, lavadoras, etc.

Para hacer sistemas interactivos hace falta [PRS+94] comprender los factores (psicológicos, ergonómicos, organizativos y sociales entre otros) que determinan como la gente trabaja y hace uso de los ordenadores y trasladar esta comprensión. Con esta comprensión, se desarrollan herramientas y técnicas para ayudar a los diseñadores a conseguir que los sistemas informáticos sean los idóneos para las actividades a las cuales se quieran aplicar. De esta forma se puede conseguir una interacción eficiente, efectiva y segura, tanto a nivel individual como de grupo.

La propiedad más valiosa de los sistemas interactivos es el soporte de la actividad humana [NL95]. Esta característica puede permitirnos realizar más rápidamente las tareas, con menos errores, menor aprendizaje, mayor calidad e incluso mayor satisfacción. Obteniendo estos beneficios se justifica el mayor coste que supone su implementación. Si, por el contrario, lo que hace es interferir con las actividades, lo único que obtendremos es la necesidad de un mayor esfuerzo para realizar la misma cantidad de trabajo o para resolver los problemas que el sistema haya causado, y más pronto o más tarde terminaremos descartando el sistema o sustituyéndolo por otro mejor diseñado.

La última generación en estilos de interfaces de usuario para sistemas interactivos son las interfaces de manipulación directa. Una interfaz de manipulación directa, término acuñado por Ben Shneiderman en 1983 [Shn83], es una interfaz donde el usuario apunta directamente a los objetos de la pantalla y los manipula, usando dispositivos como el ratón o el teclado. Se compone de una serie de objetos interactivos gráficos (ventana, región de pantalla, botón, campo de texto, barra de desplazamiento, etc.) cada uno de los cuales tiene asociado un diálogo (conjunto de operaciones que se puede ejecutar sobre el objeto). La especificación se centra en la descripción separada de cada uno de estos objetos y sus diálogos.

Las interfaces de manipulación directa son multimodales. Cada modo o estado tiene asociado un conjunto de entradas (acciones) válidas y un determinado significado para las mismas. Cada vez que se mueve el cursor para que apunte a un objeto diferente en la interfaz se produce un cambio de modo, porque una vez que el cursor enfoca el nuevo objeto, el rango de entradas válidas se reduce y su significado puede cambiar también.

En la siguiente sección se contempla un modelo que identifica los tres componentes principales del sistema interactivo.

2.3 El Humano

[CMN83]describen el Model Human Processor, que es una vista simplificada del procesamiento humano que se desarrolla durante la interactuación con el computador. Permite a los diseñadores anticipar el procesamiento que el usuario debería tener que engranar para realizar tareas elementales. El modelo incluye memoria a largo plazo y

memoria de trabajo, almacenes de imágenes visuales y auditivas y tres procesadores :

El sistema perceptual Manejador de estímulos sensoriales procedentes del mundo exterior. Determina el límite entre el entorno y el usuario. Es el responsable de las entradas desde el mundo real y está relacionado con el sistema sensorial del cuerpo. El sistema sensorial más estudiado es el visual.

El sistema motor Controla las acciones. Es el responsable de los movimientos físicos de los usuarios. En lo que concierne a la interacción hombre-máquina los más interesantes son los movimientos del ojo y de la mano. No existe un movimiento continuo sino una secuencia de movimientos elementales rápidos. Incluso mientras se piensa aparece a un nivel alto de comportamiento un movimiento continuo, como por ejemplo moviendo un icono.

El sistema cognitivo Provee del procesamiento necesario para conectar los otros dos. En interacciones simples, el sistema cognitivo sirve como interfaz entre el sistema perceptual y el sistema motor. En tareas más complejas se ocupa de comparaciones, almacenamiento de información, aprendizaje, etc.

[DFAB98] define la analogía del usuario con un sistema de procesamiento de información, pero su modelo se aproxima más al de un computador convencional. La información llega, es almacenada, se procesa y se devuelve el resultado. Establecen su modelo bajo tres componentes fundamentales: entrada-salida, memoria y procesamiento.

2.3.1 Entrada-Salida

La interacción de una persona con el mundo exterior ocurre a través de información que es recibida y enviada: entrada y salida. En una interacción con un computador el usuario recibe información que es salida de la computadora, y responde dando una entrada a la computadora (la salida de usuario es entrada de computadora y viceversa). De esta forma, el uso de los términos entrada y salida puede producir confusión.

La entrada en el usuario ocurre principalmente a través de los sentidos (vista, oído y tacto) y la salida a través del motor de control de los efectores (dedos, principalmente).

Percepción visual La información que se recibe por el sistema físico visual debe ser filtrada y enviada para procesar los elementos que nos permiten reconocer escenas coherentes, reconocer distancias relativas y diferenciar color. Como elementos importantes para el diseño de interfaces visuales efectivos describiremos la percepción de tamaño y profundidad, brillo y color.

Tamaño y profundidad La percepción del tamaño viene dada por el ángulo visual del objeto. Si un objeto se aleja paulatinamente el ángulo visual del mismo disminuye y por lo tanto se percibirá que el objeto es más pequeño. Esto realmente es una inconsistencia con la realidad, pues aunque se perciba más pequeño su tamaño real no ha cambiado. Esto se denomina ley de consistencia de tamaño y sugiere que la percepción del tamaño se debe a algún factor más que al ángulo visual.

Uno de esos factores que influyen en nuestra percepción es la profundidad. En toda escena existen una serie de claves que se pueden utilizar para determinar la posición relativa y las distancias de los objetos que se ven. Si los objetos se solapan, el objeto que se cubre de forma parcial se percibe en el fondo y por lo tanto más lejos. De forma similar el tamaño y la altura del objeto desde nuestro punto de vista nos da una clave para determinar su distancia. La tercera clave es la familiaridad, si se espera que un objeto sea de un determinado tamaño entonces podremos juzgar las distancias de forma acorde a ello.

Brillo El brillo está afectado por la luminosidad, que es la cantidad de luz emitida por un objeto. La luminosidad de un objeto depende de la cantidad de luz que recibe la superficie del objeto y sus propiedades reflectivas. El contraste está relacionado con la luminosidad, es una función de la luminosidad de un objeto y la luminosidad de su fondo.

Color El color se considera como una composición de tres elementos: matiz, intensidad y saturación. El matiz es determinado por la longitud de onda espectral de la luz. La intensidad es el brillo del color, y la saturación es la cantidad de blanco en el color. Por variación de estos dos últimos, se puede percibir alrededor de siete millones de colores diferentes. Sin embargo, el número de colores que pueden ser identificados por una persona sin entrenamiento es mucho menor.

2.3.2 La Memoria

La memoria es la segunda parte del modelo del humano en un sistema de procesamiento de información. En la mayoría de los casos, las distinciones de tipos de memoria vienen dadas por la función que realizan dentro de un nivel de procesamiento mental. Básicamente, la memoria del humano se puede clasificar en memoria sensorial, de trabajo y a largo plazo.

Memoria sensorial

Actúa como buffer de los estímulos recibidos a través de los sentidos. Permite predecir la procedencia del sonido, o un fogonazo en la oscuridad.

Memoria de Trabajo

La memoria de trabajo (STM Short Term Memory) es el conjunto de símbolos activos en un momento determinado a los que estamos prestando atención y que, por tanto, podemos manipular mediante control voluntario.

Memoria de largo plazo

La memoria de largo plazo (LTM Long Term Memory) almacena todo nuestro conocimiento. Las principales características son: gran capacidad (casi ilimitada), acceso más lento, y que las pérdidas ocurren más lentamente.

Está estructurada de modo que se puede acceder a la información mediante asociaciones y relaciones (redes semánticas).

La información de la memoria de trabajo se transfiere a la LTM a través de un proceso de memorización (recordatorio) consistente en refrescar la información (p.e. mediante repetición del estímulo).

2.3.3 El procesamiento

Hemos visto de qué forma la información se encamina desde la entrada a la salida del sistema humano y cómo se almacena. Nos falta comprobar de qué manera se procesa

y manipula en este intervalo. Esta es la parte que más diferencia los sistemas de procesamiento de información humanos de los demás, sean naturales o artificiales.

Los humanos pueden usar información para razonar y resolver problemas y pueden realizar estas actividades con información parcial o no disponible. Aunque no seamos capaces de identificar el proceso que usamos, podemos identificar los productos de este proceso, los pensamientos. Además, somos capaces de razonar de cosas sobre las que no tenemos experiencia y resolver problemas que nunca hemos visto antes. Razonar requiere cantidades de conocimiento diferentes. Algunas actividades son muy directas y el conocimiento que requieren es limitado. Otras requieren ingentes cantidades de conocimiento de diferentes fuentes, y no necesariamente relacionadas entre sí.

Tipos de razonamiento

Es el proceso por el cual usamos el conocimiento que poseemos para obtener conclusiones o inferir algo nuevo sobre el dominio de interés.

Razonamiento deductivo El razonamiento deductivo deriva una conclusión (no necesariamente verdadera) de forma lógica desde unas premisas dadas.

Razonamiento inductivo La inducción es un proceso de generalización desde casos que han sido observados previamente para inferir información sobre casos que no se han visto nunca. La inducción es un proceso útil, que se utiliza constantemente para aprender de nuestro entorno.

Razonamiento abductivo La abducción razona desde un hecho a la acción o estado que lo causa. Las personas obtienen explicaciones para los eventos de esta forma y los mantienen hasta que encuentran una evidencia o pista para soportar una teoría o explicación alternativa. Este razonamiento puede generar problemas dentro de los sistemas interactivos, dado que si un evento siempre sigue a una acción, el usuario puede inferir que el evento es causado por la acción a menos que se haga disponible para el usuario una evidencia de lo contrario. En el caso de que el evento y la acción no estén relacionados, y no se haga una indicación clara al respecto, el usuario puede sentirse confuso y puede cometer errores en el proceso.

Resolución de problemas

La resolución de problemas es el proceso de encontrar una solución a una tarea que no es familiar usando el conocimiento que se posee. Está caracterizado por la habilidad para poder adaptar la información que poseemos para aplicarla a nuevas situaciones. Aunque la información utilizada en este caso para la resolución ya es conocida, la solución parece original y creativa. Existen varias teorías a la hora de abordar la resolución de problemas.

Teoría del espacio de problema Newell y Simon propusieron centrar la resolución de problemas en el espacio de problema. El espacio de problema contiene estados de problema, y la resolución del problema implica generar estos estados utilizando operaciones de transición entre estados legales. Todo problema tiene un estado inicial y un estado objetivo o final y las personas usan los operadores para moverse del primero al último.

Una característica importante del modelo de Newell y Simon es que opera con las restricciones del sistema de procesamiento humano y así la búsqueda en el espacio de problema se limita a la capacidad de la memoria de corto plazo, y la velocidad con que la información puede ser recuperada. En el marco de trabajo del espacio de problema, la experiencia nos permite resolver problemas más fácilmente dado que podemos estructurar el espacio de problema adecuadamente y elegir los operadores eficientemente.

2.4 La computadora

La computadora puede ser definida como: *"El participante en la interacción que ejecuta un programa"*. Esta definición general puede ser aplicada a un gran número de dispositivos con los que un usuario interactúa a menudo. Tanto entre personas como con una computadora, la interacción es el proceso de transferencia de información.

Para analizar la transferencia de información entre computadora y usuario, centraremos la discusión en el proceso de visualización y cómo la tecnología actual restringe el diseño de la interfaz de usuario.

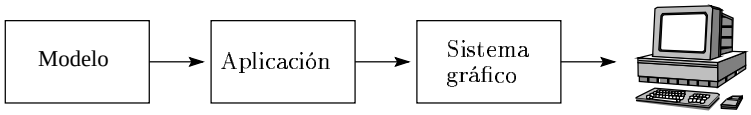

Figura 2.1: Esquema general del proceso de visualización

2.4.1 El proceso de visualización

Un sistema gráfico interactivo puede describirse a nivel conceptual formado por tres elementos [FDFH90]:

Modelo de aplicación Representa la información u objetos que serán mostrados a través del dispositivo de visualización

Programa de aplicación Crea, almacena y recupera los elementos del modelo de aplicación

Sistema gráfico Conjunto de comandos de salida gráfica que contienen tanto la descripción geométrica detallada de *qué* es lo que debe mostrarse como los atributos para mostrar *cómo* deben aparecer.

La figura 2.1 muestra un esquema de los elementos integrantes de un sistema gráfico interactivo.

El modelo de aplicación es específico de la aplicación e independiente del sistema gráfico y del dispositivo de visualización. El programa de aplicación convierte la descripción del modelo de aplicación desde la descripción interna de la geometría del modelo a llamadas o comandos del sistema gráfico para crear una imagen en el dispositivo de visualización. Para ello recoge aquella información a mostrar del modelo siguiendo algún criterio. Si la información no es geométrica, la transforma a geométrica y la describe en forma de primitivas que el sistema gráfico puede mostrar directamente, y de atributos que pueden controlar su apariencia. En dos dimensiones, las primitivas básicas son líneas, rectángulos, polígonos, círculos, elipses y texto. Entre los atributos encontramos color, estilo de línea y ancho de línea.

Cabe destacar aquí la representación de coordenadas. Con pocas excepciones, los paquetes generales de gráficas están diseñados para utilizarse con especificaciones de coordenadas cartesianas. En general, se pueden utilizar varias estructuras cartesianas para crear y desplegar una escena. Puede construirse la forma de los objetos

individuales en estructuras de coordenadas de referencia separadas, que se conocen como *coordenadas de modelado*. Una vez que se especifican las formas de objetos individuales, se pueden colocar los objetos en las posiciones adecuadas al utilizar una estructura de referencia llamada *coordenadas mundiales*. Por último, la descripción de las coordenadas mundiales se transfiere a una o más estructuras de referencia de dispositivo de salida para su visualización. Estos sistemas de coordenadas de visualización reciben el nombre de *coordenadas de dispositivo*(o coordenadas de pantalla en el caso de un monitor de vídeo). Las definiciones del modelado y de las coordenadas mundiales permiten establecer cualquier dimensión de punto flotante o entero conveniente sin obstáculos por las restricciones de un dispositivo de salida particular.

Por lo general, un sistema gráfico primero convierte las posiciones de coordenadas mundiales a coordenadas de dispositivo normalizado, en el rango de 0 a 1, antes de realizar la conversión final para especificar las coordenadas de dispositivo, Esto hace que el sistema sea independiente de los diversos dispositivos que se podrían emplear en un ordenador. Una posición inicial de coordenadas de modelado (x_{mc}, y_{mc}) se transfiere a una posición de coordenadas de dispositivo (x_{dc}, y_{dc}) mediante la secuencia [HB94]

$$(x_{mc}, y_{mc}) \Rightarrow (x_{wc}, y_{wc}) \Rightarrow (x_{nc}, y_{nc}) \Rightarrow (x_{dc}, y_{dc})$$

Las posiciones de coordenadas de modelado y mundiales en esta transformación pueden ser cualesquiera valores. Las coordenadas normalizadas satisfacen las desigualdades: $0 \leq x_{nc} \leq 1, 0 \leq y_{nc} \leq 1$; y las coordenadas de dispositivo x_{dc} e y_{dc} son enteros dentro del rango $(0, 0)$ a (x_{max}, y_{max}) para un dispositivo de salida particular.

Para ajustar las diferencias en escalas y razones de aspecto, se realiza un diagrama de coordenadas normalizadas en un área cuadrada del dispositivo de salida, de tal forma que se mantengan las proporciones adecuadas.

2.4.2 Eventos

Los eventos son actuaciones del usuario sobre los dispositivos físicos de entrada en la computadora. Estos eventos son detectados por la computadora, que actuará en función de las órdenes que tenga definidas para los mismos. En el dominio en el que el presente trabajo se sitúa, dichas órdenes corresponden a actuaciones dentro de los

componentes de la interfaz de usuario o a llamadas al núcleo de la aplicación. Las herramientas de programación que permiten el desarrollo de la interfaz de usuario pueden definir sus propios eventos, que son combinación de los eventos elementales que el usuario puede realizar sobre los dispositivos físicos de interacción.

Uno de los detalles a tener en cuenta a la hora de describir los eventos es que su funcionalidad depende del componente de la interfaz sobre el cual el usuario actúa. Esto quiere decir que para un determinado estado de la interfaz, compuesto por los estados de los componentes individuales de la misma, un evento puede tener efecto sobre varios componentes, y por lo tanto el usuario, bien a través del dispositivo apuntador o bien a través del teclado, deberá seleccionar sobre qué componente de la interfaz desea que ese evento tenga efecto.

Para ello existen eventos genéricos que no tienen una relación directa con una actuación de un dispositivo físico sino que modifican el estado de un componente particular haciendo de este el receptor de los eventos. Son eventos que se realizarán sobre el componente de la interfaz que esté seleccionado en el estado actual de la interfaz. Un ejemplo típico es el evento *"pulsar el botón SUPRIMIR"* en Windows.

Lo mismo se puede decir de otros eventos de los entornos de programación que simplifican la labor de los programadores como los aplicables sobre componentes que pueden modificar sus propiedades visuales (por ejemplo el tamaño).

Los eventos gestionados por la computadora a nivel de interfaz de usuario no tienen justificación si el usuario no ha realizado una acción mecánica sobre los dispositivos físicos que de lugar a una entrada de información externa que permita una interacción en la interfaz.

2.5 La interfaz de usuario

Myers [Mye96] define la interfaz de usuario de un programa informático como *la parte que gestiona la salida hacia el monitor y la entrada desde la persona que usa el programa* y denomina el resto del programa como la aplicación o la semántica de la aplicación.

[Mor81] define la noción de interfaz de usuario como *aquellos aspectos de sistema con los que el usuario llega a estar en contacto* y [Chi85] matiza en *un lenguaje de entrada*

para el usuario, un lenguaje de salida para la máquina y un protocolo de interacción.

También según [Mye96], el software de interfaz de usuario es frecuentemente grande, complejo y difícil de implementar, depurar y modificar. Gran cantidad del código de las aplicaciones se dedica a la interfaz de usuario y, por lo tanto, se consume una gran cantidad del tiempo de implementación. Hoy en día, las interfaces de usuario de manipulación directa son universales. Estas interfaces requieren que el programador trabaje con gráficos elaborados, múltiples formas de solicitar el mismo comando, dispositivos de entrada asíncronos múltiples, una interfaz de *"manejo libre"* donde el usuario puede realizar cualquier comando en cualquier momento y una realimentación semántica rápida, donde determinar la respuesta apropiada a las acciones del usuario requiere información especializada sobre los objetos del programa.

2.5.1 Formas de abordar su diseño

El diseño es un proceso iterativo encaminado a obtener un modelo del sistema e indicar cómo el usuario realiza sus actividades en el sistema (análisis de tareas). Este modelo se puede obtener partiendo de diferentes fuentes de información.

Los principios, guías, estándares, etc. representan el conocimiento acumulado de experiencia en el diseño de interfaces [Perl89]. Se puede obtener un buen diseño desde el conocimiento y la experiencia de los diseñadores y desde la forma en que estos aplican su conocimiento [PRS+94]. Las guías de diseño pueden ayudar a proporcionar un marco de trabajo en el que se puede guiar a los diseñadores en la toma de decisiones.

Podemos guiar el diseño de las interfaces de usuario desde el punto de vista del tipo de funcionamiento que se espera para la misma. De esta forma distinguiremos distintos tipos de interfaces de usuario dependiendo del método de aprendizaje del sistema y del entorno de trabajo y necesidades de comunicación de los usuarios del mismo.

A la hora de diseñar la interfaz de usuario debemos tener en cuenta al protagonista de la misma, el usuario, y guiar el diseño anteponiendo las características perceptivas, cognitivas y estéticas que éste posee.

Tipos de interfaces en el diseño

Dentro de las opciones de diseño de los Interfaces de Usuario podemos distinguir algunas[Mac96] que se pueden explorar cuando se intenta diseñar una interfaz.

Interfaces demostracionales Las interfaces de manipulación directa, aunque son muy utilizadas, tienen ciertos problemas:

- La realización de tareas comunes, repetitivas o muy relacionadas es tediosa.
- No permiten abstraer o generalizar especificaciones.
- Para programar una DMI, el usuario tiene que estar al tanto de ciertos lenguajes y técnicas de programación, y éstas en general son complejas.

Myers señaló estos defectos en las Interfaces de Manipulación directa e introdujo la noción de Interfaces Demostracionales [Mye92]. Las interfaces demostracionales permiten al usuario proporcionar ejemplos concretos característicos de las acciones que se pretenden realizar, y completar el resto de las acciones con poca o ninguna realimentación desde el usuario. La idea de utilizar la demostración en la programación no es un concepto nuevo [Hal84]. Las facilidades del uso de la demostración se ha estudiado en aplicaciones como editores de texto, editores gráficos u hojas de cálculo.

Pueden verse distintas clases de interfaces demostracionales en [Mye92]. El hecho de que no se encuentre ningun sistema comercial que utilice interfaces demostracionales muestra que existen ciertos problemas asociados a su utilización. Son difíciles de utilizar cuando el usuario tiene que especificar exactamente lo que necesita.

Interfaces Inteligentes La programación automática fue un tema candente en Inteligencia Artificial en los primeros años de los 70. En la programación automática, las computadoras utilizan inteligencia (o heurísticas) para generar programas a partir de un conjunto de especificaciones simples. El principal foco de investigación, en ese momento, era generar programas automáticamente desde pares de entrada/salida o ejemplos de ejecuciones de programas. Posteriormente, esas técnicas se volvieron a usar en el contexto de ciertos dominios limitados como las interfaces de usuario [ST91].

Las heurísticas son usadas en algunas interfaces para guiar a los usuarios y realizar las tareas difíciles de especificar utilizando aproximaciones de manipulación directa convencional. Las heurísticas son una técnica de resolución de problemas en que la solución más apropiada se elige mediante reglas. Las ventajas generales de estos sistemas son que pueden ahorrar algún tiempo al usuario dado que no se tienen que especificar todos los detalles, y su aprendizaje puede ser más fácil dado que el sistema realiza parte del trabajo. Sin embargo, pueden realizar acciones incorrectas (que pueden no ser percibidas por el usuario), por las cuales el usuario puede no entender por qué el sistema realiza una acción diferente o cómo puede ser realizada la acción que se pretende. Los usuarios entonces sienten que el sistema es impredecible y que no poseen el control. La investigación en esta área se centra en el desarrollo de herramientas e interfaces que solucionen estos problemas, y las formas de manejar las heurísticas que serán usadas en el sistema.

Interfaces Distribuidas y Colaborativas El paradigma de programación Cliente-Servidor ha llegado a ser popular con el desarrollo de los ordenadores personales y redes de área local. En este modelo, uno o más clientes y uno o más servidores, a través de un sistema operativo subyacente y sistemas de comunicación entre procesos, forman un sistema compuesto que permite computación, análisis y presentación distribuida. Algunos de los avances en tecnología Cliente-Servidor y su impacto en las interfaces de usuario pueden verse en [Shi92]. Dado que un sistema cliente-servidor consiste en múltiples clientes, pueden existir múltiples interfaces de usuario en el sistema, pero cada cliente tiene una única y consistente interfaz de usuario.

Las interfaces de usuario distribuidas tienen un área de investigación más general.

- Interfaces de usuario en entornos de computación distribuidos (o entornos colaborativos).
- Interfaces de usuario con componentes distribuidos.

Las interfaces distribuidas se necesitan cuando una aplicación y una interfaz no utilizan memoria compartida. Ejemplos de estos son muy comunes en aplicaciones de bases de datos, donde las bases de datos residen en máquinas aisladas y las interfaces utilizan las capacidades de las estaciones de trabajo gráficas. Además de todas las ventajas de los entornos de computación distribuida [Shi92], esta aproximación tiene dos ventajas adicionales:

- Aislamiento entre aplicación e interfaz.
- Adaptación personal a interfaces individuales.

Dentro de las interfaces colaborativas está el Computer Support in Collaborative Work (CSCW) [Gru91]. La llegada de los sistemas multimedia ha centrado el interés en este área. Por una parte, existen intentos de construir herramientas que permitirían a los usuarios de diversos campos de conocimiento sentarse juntos y diseñar o construir interfaces colaborativamente.

2.5.2 Diseño Visual

En esta sección se introducen los conceptos básicos de la teoría de comunicación visual. También se establece la relación entre estos conceptos y su importancia en algunas características de la percepción, o interpretación de la información visual por parte de los usuarios, de la interfaz.

Algunas definiciones en la teoría de la comunicación visual son las siguientes:

Comunicación Proceso completo por el cual el comportamiento de una entidad llega a verse afectado por el de otra a través del intercambio recíproco de mensajes o signos sobre un canal físico mediático.

Diseño Visual Intenta resolver problemas de comunicación de una forma que es a la vez efectiva funcionalmente y agradable estéticamente.

Semiótica Teoría y práctica general de signos [Pie31] y [Mor38]

Lenguaje visual Características visuales (forma, tamaño, posición, orientación, color, textura, etc.) de un conjunto particular de elementos de diseño (puntos, líneas, planos, volúmenes, etc.) y la forma en que se relacionan unos con otros (balanceo, ritmo, estructura, proporción, etc) en la resolución de un determinado problema de comunicación.

Sistema Lenguaje Define un universo de posibles signos y un conjunto de reglas para usarlos.

El objetivo del diseño orientado a comunicación es desarrollar un mensaje que pueda ser transmitido exactamente e interpretado correctamente, y que producirá el comportamiento deseado después de que haya sido entendido por su receptor.

Se entiende como factores humanos aquellas características de la percepción humana que modifican las interpretaciones que el usuario realiza de una interfaz de usuario cuando la observa. Existen una serie de conceptos visuales que el usuario interpreta de una forma particular debido al procesamiento mental de la información visual que recibe. De hecho, estos conceptos han sido muy utilizados por diseñadores gráficos y publicistas a lo largo del tiempo.

Se introducen ahora los antecedentes relativos a las interfaces de usuario gráficos, y a continuación los factores humanos principales que afectan a su interpretación.

Antecedentes La llegada de la tecnología GUI ha abierto nuevos grados de libertad en el uso del color, tipografía e imágenes. Sin embargo, mientras que el nivel técnico de producción fue impresionante, la calidad estética deja mucho que desear. Irónicamente, dada la naturaleza gráfica de la revolución de los GUI, las imágenes son uno de los principales problemas. La cantidad de iconos necesarios para satisfacer la demanda de aplicaciones y usuarios, combinada con la necesidad de soportar monitores de baja resolución y 16 colores han permitido la proliferación de imágenes de baja calidad que han sacrificado el ancho de banda valorable de comunicación en un propósito bien intencionado de divertir y lucir.

Por otro lado, los toolkits de interfaz de usuario existentes proveen muy poco soporte al diseño visual orientado a la comunicación. De hecho, la mayoría de los toolkits imponen restricciones de diseño innecesarias como un efecto lateral de su propia implementación o arquitectura interna.

Otro problema importante es el desorden en la colocación de controles en las ventanas y los cuadros de diálogo. De hecho, la interfaz típica de las aplicaciones fue probablemente estructurada de forma más efectiva en los días de los dispositivos de visualización orientados a caracteres, dado que el número limitado de posiciones en una pantalla en modo carácter tenía que ser manejado cuidadosamente.

A continuación, vamos a ver algunas de las reglas definidas a nivel de diseño visual para lograr una correcta comunicación haciendo especial hincapié en aquellas que pueden ser formuladas matemáticamente o existe algún método de evaluación cuantificable. Existen otras muchas reglas de diseño que no poseen esta característica de valuabilidad y precisamente ahí reside su defecto o virtud. Al ser reglas absolutamente subjetivas, el usuario debe decidir su utilización o no dependiendo del significado que pretenda

dar a los signos o componentes presentes en la comunicación.

Regularización La regularidad reduce información por repetición de elementos de acuerdo a una regla discernible, un principio o un ritmo. La percepción humana y la memoria operan más eficientemente sobre un estímulo regularizado, dado que la complejidad visual de la pantalla se reduce mientras su estructura se procesa. La predictibilidad de un esquema regular permite a la persona que lo ve una percepción más fácil del área de interés cuando realiza una comparación o responde una pregunta. La regularidad introduce también beneficios a nivel estético. Esto es evidente en, por ejemplo, el efecto en decoración con el uso de esquemas repetitivos.

La regularidad puede ser también lograda por alineamiento o reflejo de elementos a lo largo de ejes comunes, por estandarización o repetición de tamaños y espaciado de componentes o por reducción de componentes a formas geométricas básicas donde sea posible.

Un diseño efectivo logra un equilibrio entre contraste y regularidad. Además establece un ritmo predecible.

Establecer un esquema simplifica el diseño al mover la experiencia de la persona que ve el diseño a un nivel de abstracción más alto. De este modo, una serie de rectángulos blancos y negros llegan a ser un tablero de damas cuando se colocan adecuadamente. Los elementos de diseño deben ser regularizados en muchos niveles simultáneamente para producir este efecto.

Cualquier irregularidad será interpretada como significante por el usuario, el cual alegremente le dará un significado incluso en aquellas ocasiones en las cuales no se intentaba darle ese significado. Si se produce la regularización de los elementos de diseño no críticos durante el trabajo, se podrá atraer la atención del usuario introduciendo irregularidades obvias en los momentos en que se desee hacer una distinción.

Cardinalidad de la agrupación de componentes La cardinalidad de los componentes que están agrupados de alguna forma física o lógica tiene relevancia en relación con la capacidad de la memoria de trabajo de los usuarios. Como característica importante dentro de las restricciones que la capacidad de procesamiento determina en el ser humano está la capacidad de la memoria de trabajo, que debe ser tenida en cuenta a la hora de determinar el número de elementos que forman parte de un grupo

dentro de la interfaz de usuario.

Simetría La simetría, en todas sus formas, tiene un atractivo estético universal reflejado en su connotación más general de *"belleza como resultado de equilibrio o colocación armoniosa"*. El ratio 1:1, que forma la base de toda simetría, es fácilmente reconocible e inherentemente satisfecho. En cualquier sitio que una forma se repita, ya sea en traslación (repetición), rotación o reflejo (imagen espejo), la simetría actúa para unificar aquellas partes de la configuración que comparten características formales similares. La naciente popularidad y eficacia de los formatos asimétricos en los diseños modernos reflejan no una negación de las cualidades inherentes a la simetría, sino la necesidad de presentar incrementalmente información compleja de una manera orientada a tarea. La simetría queda como una herramienta potente, particularmente cuando el objetivo de la comunicación depende del equilibrio, orden y simplicidad.

La simetría asegura equilibrio y organización, algunas veces a costa del interés visual. Afortunadamente, el objetivo del diseño de una interfaz efectiva no es entretener o excitar, sino presentar la información eficazmente y no de forma intrusiva. Así, el carácter relajante del formato simétrico es, en la mayoría de los casos, perfectamente apropiado. Dada la facilidad con que puede ser aplicado, la técnica es un punto de partida de valor incalculable para un diseño efectivo.

Debido a que la simetría contribuye a un buen diseño en varias dimensiones a la vez, su papel histórico no debería sorprender. Ya los griegos utilizaban un concepto de simetría basándose en que cada elemento de un edificio debe estar relacionado con el resto de elementos. Las construcciones sacras de la Grecia clásica estaban basadas en cánones matemáticos elaborados que relacionaban el ancho, alto y espaciado exterior e interior de los componentes con los demás componentes. El canon es paramétrico. El ratio más famoso es el de la sección áurea. De hecho, los conceptos del diseño asimétrico son un fenómeno muy reciente que encuentra una extensa aceptación en medios impresos solo en el siglo XX. Al contrario que la complejidad del formato asimétrico efectivo, mantener la simetría es un proceso relativamente directo que puede ser usado para asegurar un equilibrio adecuado en casi cualquier diseño.

Alineamiento El alineamiento es uno de los medios más importantes de establecer relaciones entre elementos. Los objetos alineados crean una fuerte atracción entre ellos incluso cuando están separados por grandes distancias. El ojo exhibe una ten-

dencia natural hacia la percepción de estructuras regulares. El observador intentará dar sentido a una imagen por división en regiones a lo largo de los ejes estructurales principales. La información adicional introducida por variaciones accidentales o irrelevantes en la colocación o el tamaño de los elementos de la interfaz es simplemente ruido visual que inhibe la comunicación.

Ajuste óptico El diseño visual está basado en fenómenos perceptuales más que en fenómenos físicos. El alineamiento visual depende de forma crítica de un ajuste óptico para compensar las diferencias en forma y contorno entre los elementos que están alineados. El ajuste óptico es un fenómeno que se aplica no solo al alineamiento sino también al escalado y el espaciado. Los elementos curvos, en general, deben proyectarse ligeramente más lejanos al borde que los elementos lineales, mientras que los ángulos más agudos deben proyectarse incluso más para lograr el mismo efecto. Los círculos deben ser un poco más largos que los cuadrados y los rombos un poco más largos que los círculos. Este fenómeno aparece frecuentemente con el diseño de iconos.

El diseño visual efectivo explica las cualidades ópticas de todos los elementos de diseño y provee compensación donde sea necesario. No hay, desafortunadamente, ningún método seguro para determinar el grado de ajuste necesario de forma mecánica. Una observación cuidadosa y una práctica extensa son la única manera de desarrollar el nivel necesario de diseño. En cualquier caso, los ejes paralelos más predominantes en la componente de la interfaz deberían ser usados para establecer la relación de alineamiento.

El objetivo del diseño efectivo es crear relaciones visuales aparentes entre elementos relacionados conceptualmente. El ajuste óptico asegura que estas relaciones se logren cuando, y solo cuando, las relaciones conceptuales se realizan apropiadamente.

Espacio negativo Los diseñadores en cualquier ámbito son frecuentemente presionados por sus clientes para incluir tanta información como sea posible en todas las páginas o pantallas. Sin embargo, las regiones aparentemente vacías juegan un importante papel, el de dirigir la atención del observador hacia las regiones en donde se encuentra la información importante y permitir a la estructura global de la composición asumir una configuración con significado. Sin espacio negativo, que es simplemente otra forma de describir las cualidades del fondo en que aparecen las

figuras, no puede existir una estructura global con significado.

El espacio negativo, o espacio en blanco, es necesario para la integración de figuras con el fondo. El ojo debe ser dirigido hacia las claves necesarias para localizar la información de interés. Una cuidadosa colocación del espacio en blanco es la técnica más efectiva para lograr este objetivo.

La segregación espacial es la variable perceptual más potente. Es también prácticamente la más costosa y por ello debe usarse juiciosamente. El espacio en blanco puede usarse efectivamente para llamar la atención sobre algún elemento importante de la pantalla cuando no se puede utilizar un tamaño grande o un contraste elevado. Cuando existe código redundante, la necesidad de segregación espacial disminuye, aunque no suele desaparecer totalmente. La cantidad exacta de espacio para colocar las estructuras de pantalla es siempre una cuestión de sentido común.

2.5.3 Su evaluación

Los métodos de evaluación son modelos que predicen aspectos de usabilidad a partir de la especificación del diseño. En [Smi97] se recogen una serie de definiciones asociadas a la usabilidad de las interfaces de usuario.

Según del Diccionario de Computación de IBM (1993) [IBM93], utilidad es la capacidad de un sistema, programa o dispositivo para realizar las funciones para las cuales fué diseñado.

La Organización Internacional de Estándares (ISO) en el ISO 9241-11.3 [ISOa] define:

Efectividad es la precisión y completitud con que los objetivos logran objetivos específicos.

Eficiencia es la precisión y completitud de objetivos en relación a los recursos utilizados.

Satisfacción es la comodidad y aceptabilidad del sistema.

En [PRS+94] se indica que la evaluación está relacionada con la obtención de datos sobre la usabilidad de un diseño o producto por un grupo de usuarios específicos para una actividad particular con un entorno o contexto de trabajo específico. A la hora de evaluar un sistema, e independientemente del tipo de evaluación que se realice,

Descripción	Fuente
"Ofrece funcionalidad en cada una de las formas con las que los usuarios podrán manejar y explotar el sistema sin tener en cuenta sus capacidades y niveles"	Eason, 1988 [Eas88]
"Cualidad de un sistema, programa o dispositivo que le permite ser fácilmente entendible y convenientemente aplicado por el usuario"	Diccionario de Computación de IBM, 1993 [IBM93]
"... efecto, capacidad de aprendizaje, flexibilidad y actitud	Shackel, 1986 [Sha86]
"Tiene múltiples componentes y es tradicionalmente asociado con cinco factores de usabilidad: capacidad de aprendizaje, eficiencia, retención, errores y satisfacción "	Nielsen, 1993 [Nie93]
" Un conjunto de atributos software que indican el esfuerzo necesario por el uso y sobre las valoraciones individuales de cada uso por un conjunto de usuarios establecido o implícito"	ISO/IEC 9126, 1992 [ISOb]
"Efectividad, eficiencia y satisfacción con las que usuarios específicos pueden conseguir objetivos específicos en entornos particulares"	ISO 9241, 1993 [ISOa]

Tabla 2.1: Definiciones de usabilidad. De más general a más específica

debe considerarse lo siguiente:

- Las características de los usuarios (o posibles usuarios) del producto que toman parte en la evaluación
- Los tipos de actividades o posibles actividades que los usuarios realizarán.
- El entorno de estudio, que puede ir desde una situación de laboratorio controlado a un entorno natural de trabajo (este sería considerado un estudio de campo).
- La naturaleza del artefacto a ser evaluado, que puede ir desde un conjunto de bocetos a un prototipo software o a un producto final totalmente desarrollado.

Incluso en evaluaciones predictivas, donde los expertos intentan predecir la usabilidad de un producto sin involucrar directamente a los usuarios, los evaluadores deben tener en cuenta esos cuatro aspectos.

Métodos de evaluación cualitativos

Existen múltiples métodos de recogida y análisis de información acerca de la interfaz de usuario. La elección de un método u otro no depende solo de cual es la respuesta que se quiere conocer sino de múltiples factores que pueden resumirse en *"¿Cuánto cuesta y qué obtendremos de su realización?"*. Los métodos no son totalmente independientes sino que se solapan en cuanto a las actividades que desarrollan.

Dentro de los métodos utilizados para la evaluación cabe destacar los siguientes:

Evaluación interpretativa Permite a los diseñadores entender mejor cómo los usuarios usan sistemas en su entorno natural y cómo el uso de estos sistemas se integran con otras actividades. Se intenta recoger los datos causando la mínima interrupción a los usuarios. Dentro de este método de evaluación podemos distinguir la evaluación cooperativa, participativa, contextual y la etnográfica. La evaluación cooperativa proporciona procedimientos para que el diseñador pueda trabajar con un conjunto de usuarios del software e intenta detectar problemas mediante prototipos y simulaciones. La evaluación participativa es más abierta y deja un mayor control a los usuarios fomentando su participación. En la evaluación contextual los diseñadores y usuarios colaboran en la identificación y comprensión de los problemas de usabilidad

en el entorno de trabajo normal del usuario. La evaluación etnográfica es asimilada de la antropología y se basa en colocar a los diseñadores en el sitio de los usuarios a los que se pretende evaluar.

Evaluación predictiva Pretende predecir con qué problemas se encontrarán los usuarios al utilizar la tecnología sin testearla con los mismos. Se utilizan técnicas de modelado psicológico, como el análisis de tecleo o la revisión por expertos del diseño que por su experiencia pueden predecir problemas que el usuario típico podría encontrar. El énfasis se realiza en el diálogo de interacción entre el usuario y el sistema. Usando técnicas de simulación, un experto simula el comportamiento de un usuario con menos experiencia e intenta describir los problemas de usabilidad que dicho usuario encontraría. Dentro de la revisión de tareas se encuentra la evaluación heurística, donde los revisores examinan el sistema como en una revisión general o usando simulación, pero su inspección se guía por un conjunto de heurísticas de alto nivel.

Basados en el diseño gráfico Dentro de los métodos de comprobación de usabilidad, debemos distinguir la contribución del diseño gráfico. El diseño gráfico es la disciplina de la comunicación visual efectiva [RGBG95]. Mientras las guías y la heurística proveen soporte a los diseñadores de interfaces, el diseño gráfico efectivo puede ayudar a su implementación de forma integral. En términos de criterios de usabilidad, el diseño gráfico puede realizar una contribución valuable: satisfacción.

Marcus [Mar92] da una introducción a la aplicación del diseño gráfico en el desarrollo de interfaces de usuario. Introduce el término *"lenguaje visible"*, al que se refieren como *"todas las técnicas gráficas utilizadas para comunicar el mensaje o contenido"*. Este término incluye: layout (formatos, proporciones y rejillas), tipografía (selección de fuentes y formatos de letra),color y textura, iconografía (uso de signos, iconos y símbolos), animación, secuenciación(la aproximación más general al guión), sonido e identidad Visual (reglas para el mantenimiento de la consistencia).

Métodos de evaluación cuantitativa

Ya se ha comentado la importancia de la componente visual de las interfaces de usuario en las aplicaciones. Se ha indicado también la necesidad de una evaluación de

la misma y se han mostrado algunas de las técnicas de evaluación existentes basadas, sobre todo, en examen por parte de un humano de dicha componente visual.

Sin embargo, deben buscarse también técnicas de evaluación que minimizen la subjetividad del estudio de usabilidad y proporcionen métodos automatizables para el mismo proporcionando, entre otras cosas, un sistema fiable de comparación entre diseño y medida cuantitativa de su efectividad.

[FDFH90] y [Mar92] advierten de la tendencia entre los diseñadores de enfatizar los elementos estéticos de la interfaz de usuario, dado que puede degradar la usabilidad del mismo. Sin embargo, la aplicación de los conceptos estéticos pueden ayudar en una serie de parámetros tales como:

Aceptabilidad [Tra97] y [KK95] muestran una alta correlación entre las percepciones estéticas de la interfaz de usuario y la usabilidad.

Motivación [Toh98] indica que las composiciones agradables estéticamente tienen efecto sobre la motivación de los estudiantes en el aprendizaje basado en el modelo ARCS [Ngo94].

Capacidad de aprendizaje [Asp91] fundamenta que buenos diseños gráficos y pantallas atractivas contribuyen en la transferencia de información.

Comprensibilidad [Tul81] afirma que el rediseño de una pantalla clave de un sistema para testear líneas telefónicas provocó un 40% de reducción en el tiempo requerido por el usuario para interpretar la pantalla.

Productividad [KG83] indican que el rediseño de una serie de pantallas provocó una reducción del 25% en el tiempo total de procesamiento y una reducción del 25% en el ratio de error.

Aunque el conocimiento de las tareas y habilidades de los usuarios es clave para el diseño de pantallas efectivas, una métrica automatizable y objetiva de diseño de pantalla es una ayuda esencial. [Tul88] desarrolló cuatro métricas para salidas alfanuméricas. [SW84] propuso una medida objetiva para marcar propiedades espaciales de salidas alfanuméricas. [Sea93] desarrolló una métrica dependiente de tarea llamada *"layout appropiateness"* para asegurar que la colocación espacial está en armonía con las tareas del usuario.

[NT00] desarrolló catorce medidas estéticas cuantitativas para dispositivos gráficos reuniendo las guías y la información relacionada con la construcción de salidas gráficas correctamente definidas.

2.6 Conclusiones del capítulo

Parece claro que existe una relación de interdependencia entre la interfaz de usuario de una computadora y el humano que la utiliza. La misma naturaleza del usuario impone una serie de restricciones dadas por su capacidad de percepción y de interpretación de la información que recibe a través de sus sentidos. De la misma forma, la naturaleza de la computadora posee también una serie de restricciones dadas por las limitaciones de la tecnología que utiliza. Puestos que ambos deben coexistir para realizar tareas juntos deben amoldarse en función de sus propias limitaciones.

Las posibilidades de interacción desde el usuario hacia la computadora son finitas y vienen dadas por la tecnología disponible. Las capacidades físicas del usuario también son finitas y deben adecuarse a ellas los sistemas de salida de la computadora y los de visualización de información.

Todos los estudios acerca de la usabilidad de las interfaces de usuario por parte de los mismos se basan en las percepciones y, por lo tanto, en las capacidades físicas y mentales de los usuarios a la hora de trabajar con las computadores a través de sus interfaces de usuario.

El proceso de validación que pueda realizarse sobre la interfaz de usuario se centra en dos tipos principalmente. Por una parte, validación de la interfaz mediante expertos y usuarios que producen un informe cualitativo acerca de la usabilidad de la interfaz. Por otro lado, validación de la interfaz mediante el estudio de parámetros estéticos cuantificables, aunque no se pueda obtener ningún tipo de conclusión acerca de la validez de dichos parámetros estéticos con respecto a la intencionalidad comunicativa que el diseñador pretendía en la interfaz de usuario.

En el presente trabajo se introducirá un sistema de representación de la interfaz de usuario que permita analizar la componente visual del mismo tanto dentro del proceso de interacción como del proceso de percepción del usuario. Se representará tanto la apariencia visual de la interfaz de usuario como los eventos que la misma tiene relacionados para la correcta realización del proceso de interacción entre ambos.

Para poder relacionar los sistemas de representación de la interfaz de usuario con este trabajo, se introducen en el capítulo siguiente los modelos más utilizados de diseño de interfaces de usuario y se establece su adecuación a la representación de la componente visual de la interfaz de usuario.

Capítulo 3

Estado actual de la representación de la componente visual

En el presente capítulo se aborda la descripción de los pasos necesarios a la hora de llevar a cabo el proceso de diseño de una interfaz de usuario, mediante la descripción de las etapas conceptuales que es necesario estudiar. No es objeto de este trabajo la descripción detallada de cada uno de los modelos y herramientas que existen. Únicamente se realiza una indicación de los pasos que contemplan en el diseño de interfaces, descubriendo de este modo las lagunas existentes en los modelos y herramientas a la hora de realizar el proceso de diseño de una interfaz de usuario.

3.1 Pasos en el diseño de interfaces

3.1.1 Modelo Mental(MM)

Un elemento importante a la hora de realizar la especificación de un sistema es el modelo mental que se construye del mismo durante la fase de análisis, en base a un modelo conceptual que describe los distintos elementos que pueden aparecer en un sistema de este tipo. Los modelos de referencia conceptuales permiten estudiar los

componentes que debe poseer un sistema interactivo, sirviendo de guía durante el proceso de diseño del software. Los modelos mentales se desarrollaron para representar los aspectos dinámicos de la actividad cognitiva. Su descripción es díficil; de hecho, en IPO se ha manifestado de múltiples formas. Una definición aceptada es la proporcionada por Donald Norman[Nor88]: *el modelo que las personas tienen de sí mismas, los otros, el entorno y las cosas que con ellos interactúan. Las personas forman modelos mentales a través de la experiencia, el entrenamiento y el aprendizaje.* Se construyen modelos mentales del mundo con objeto de realizar predicciones sobre un evento externo antes de llevar a cabo una acción. Es importante resaltar [Nor83] que los modelos mentales suelen ser incompletos, inestables y frecuentemente confusos.

3.1.2 Modelo de Tareas(MT)

Los modelos cognitivos describen, de forma abstracta, el conocimiento que el usuario debe poseer acerca del sistema para su correcta utilización. Son representaciones de las tareas de usuario asociadas a su modelo mental del trabajo que tiene que realizar. Incorporan el concepto de *meta* (objetivos a alcanzar) y de *tarea* (medios para alcanzar la meta). Para el análisis de requisitos de la interfaz realizan un *análisis de tareas*. El análisis de tareas proporciona información de cómo los usuarios llevan a cabo normalmente su trabajo y sirve para identificar la funcionalidad que debería proporcionar la interfaz, representada mediante el conjunto de tareas que deben llevar a cabo para conseguir sus objetivos. Su utilidad viene dada porque refleja las tareas que el usuario realiza y, por lo tanto, debe existir una relación biunívoca entre estas tareas a realizar por el usuario y las que debe contemplar la interfaz de usuario. Es un documento importante a la hora de validar la completitud de la interfaz.

3.1.3 Modelo de Aplicación(MA)

Está presente en todos los modelos de interfaz y sirve como elemento de distinción entre la interfaz de usuario y la aplicación subyacente. Representa y describe las propiedades del código funcional que son relevantes para la interfaz. También se refiere al enfoque, desde el punto de vista de la arquitectura de software, que va a utilizar la interfaz. Muchos lenguajes de especificación que estudiaremos se basan en el concepto de *Agente* que se comunica directamente con el usuario (también llamado

Objeto Interactivo). Los agentes son objetos especializados que poseen un estado y reaccionan ante eventos causados por el usuario, el código funcional, o por otros agentes. Así, la especificación conceptual de un agente consta de dos partes[HC95b]:

- el estado del agente y las operaciones aplicables sobre dicho estado (la semántica del agente)
- la secuencia de eventos a los que el agente puede responder (la sintaxis del agente)

El sistema interactivo es un conjunto de agentes independientes pero cooperativos. Existen condiciones que definen cuando se comunican los agentes, quien controla el diálogo y que información se transmiten. El Modelo de Aplicación es importante porque genera una gran dependencia entre la implementación final del sistema y la identificación de componentes a nivel de modelo abstracto de presentación (que veremos posteriormente). La definición de los componentes abstractos de presentación debe ser siempre coherente con el Modelo de Aplicación definido.

3.1.4 Modelo de diálogo(MD)(del comportamiento interactivo)

Estos modelos buscan una correcta especificación del diálogo (descripción de la comunicación de cada participante). Representan las tareas que el usuario realiza a través de la interfaz. La diferencia que existe con el Modelo de Tareas es que se encuentran a diferentes niveles de abstracción. Mientras que el Modelo de Tareas está a nivel de tareas de usuario pertenecientes a su modelo mental, el Modelo de Diálogo representa las tareas que el usuario realiza a nivel de Modelo de Aplicación. Un modelo de diálogo describe la interfaz en términos de estados y comandos que transforman estados[Jac86]. Un sistema comienza en un estado inicial s_0 y alcanza nuevos estados $s \in S$ como consecuencia de la transición de estados producida por los comandos. Un estado está compuesto por atributos (el cursor, las dimensiones de pantalla, etc), algunos de los cuales son visibles. Cada estado se relaciona con un conjunto de atributos perceptibles mediante una función de rendering. Los comandos se invocan mediante una secuencia (traza) concreta de entradas (pulsaciones de teclas, clicks del ratón, etc.) que corresponden a acciones (eventos) del usuario. Todo sistema está diseñado para soportar un conjunto de tareas $t \in T$. Cada tarea se implementa mediante el

conjunto de todas las posibles secuencias de comandos que le permiten alcanzar sus metas (incluso las secuencias originadas por un error), partiendo siempre del estado inicial s_0. De este modo, los sistemas siempre son secuenciales. En [HB93] las tareas, denominadas *objetivos*, se asocian con conjuntos de eventos semiordenados que se expresan mediante pares pre y post-condición.

3.1.5 Modelo arquitectónico (de presentación)

Representa la estructura del sistema interactivo, reflejando los aspectos relevantes del sistema (apariencia, eventos, comunicación, etc.). Respecto a la comunicación, debe tratar tanto con la entrada (comunicación usuario-sistema) como con la salida (comunicación sistema-usuario). Estos dos canales de comunicación, que se suelen denominar *activación* y *rendering* respectivamente, son básicamente diferentes ya que la activación es basada en eventos mientras que el rendering es basado en estados. Los modelos de presentación describen la arquitectura del sistema mediante la definición de los componentes que forman la interfaz y su comportamiento. Realizan una descripción modular que facilita la composición de componentes simples para la descripción de elementos más complejos. Sus lenguajes de especificación suelen trabajar con interadores [DH93][HC95b]. Los interadores son agentes que interactúan con el usuario; están formados por un agente junto a su presentación visual. De este modo, un interador posee tres componentes: estado actual, eventos para su manipulación y una visua-lización que muestra al usuario su estado y acepta su intervención para invocar operaciones del agente. En este caso, el agente corresponde a la especificación abstracta del interador; define la organización lógica de la información presentada en la interfaz y supone el punto inicial para la determinación de su presentación. Los interadores pueden combinarse para formar otros más complejos; la propia interfaz puede modelarse como un interador *a nivel de sistema*.

Es posible dividir los modelos arquitectónicos en dos grupos:

- modelos arquitectónicos de orden superior (Modelos de Presentación ABSTRACTOS (MPA)): la semántica de sus componentes no incluye aspectos de implementación. No existe ninguna referencia a la visualización que el componente debe tener y es totalmente independiente del entorno donde se va a implementar.

- modelos arquitectónicos de implementación (Modelos de presentación CON-

CRETOS (MPC)): contienen detalles sobre la implementación de cada componente. Son dependientes del entorno de implementación.

Rendering

El término rendering engloba cualquier forma de comunicación dirigida desde el código funcional hacia el usuario. Atendiendo a su función (semántica) en la aplicación, puede clasificarse en [BPLM98]:

- Rendering de Diálogo: rendering para informar al usuario de la evolución de la interacción que está teniendo lugar.
  - de Activación: acciones que son permitidas por el sistema en un momento dado (desactivando las no permitidas, por ejemplo)
  - de Navegación: posibilidades de navegación entre ventanas (ocultando o deshabilitando las ventanas no permitidas en un momento dado)
  - Estado del Diálogo: para indicar que el usuario está en un determinado estado de interacción (cambiando, por ejemplo, la forma del ratón), mostrando el número de veces que ha accedido a una función determinada, el número de intentos posibles para autentificación de password, etc.
- Rendering Semántico: corresponde a la interacción con los objetos del sistema
  - Estado del Sistema: visualización de atributos de los objetos de la aplicación
  - Notificación: sobre cambios en el funcionamiento del sistema debido a causas independientes de la interfaz.
  - Informes de Error: por errores en la propia aplicación.

Todas las aproximaciones para la especificación y diseño de sistemas interactivos deben proporcionar alguna forma de especificar el rendering.

3.1.6 Modelo de Composición

Los modelos de composición permiten representar la colocación espacial de los componentes descritos en los modelos arquitectónicos. Es posible dividirlos en dos grupos:

- Modelos de Composición ABSTRACTOS (MCA): permiten la representación abstracta de los componentes del Modelo de Presentación Concreto a nivel de colocación en el dispositivo. Con este tipo de modelo obtenemos una representación de componentes independiente de la plataforma y donde es posible definir las reglas espaciales entre los componentes concretos de la interfaz, y su comportamiento.

- Modelos de Composición CONCRETOS (MCC): permiten representar la colocación espacial de los componentes descritos en el modelo concreto de presentación sobre el dispositivo de visualización. Algunas herramientas de especificación de interfaces disponen para ello de algoritmos, o utilizan sistemas basados en conocimiento.

3.2 Estudio de los modelos

El objetivo del presente trabajo es crear un sistema de representación abstracta, mediante el cual y con independencia del sistema de desarrollo y explotación elegido o elegible para una aplicación informática, se pueda realizar una definición de la apariencia visual y del comportamiento de la interfaz.

La apariencia visual de la interfaz está compuesta de la geometría y atributos que determinan la visualización de la geometría para cada uno de los componentes de la interfaz. Además existe una composición de los componentes en cuanto a la colocación y restricciones topológicas que existen entre ellos.

La representación de la apariencia visual de la interfaz (previa a su implementación) debe contener los siguientes elementos:

- La representación abstracta de la apariencia visual de los componentes individuales (geometría y atributos).

- La representación abstracta de la composición visual de los componentes de la interfaz con el fin de lograr una interfaz de usuario visual final.

- La representación abstracta del diálogo entre los componentes de la interfaz. Se representan el componente que recibe el diálogo del usuario (acción del usuario) y los cambios que provocan en los restantes componentes de la interfaz.

Las representaciones abstractas tienen relación con diferentes fases de diseño. Por un lado, la representación de la apariencia visual se encuentra en el modelo de presentación abstracta, el cual incluye habitualmente una descripción funcional del componente. La representación abstracta de la composición está en el modelo abstracto de composición, pero algunos modelos lo incluyen en el Modelo de Presentación Abstracto, definiendo directamente las relaciones espaciales con el resto de componentes. La representación abstracta del diálogo puede producirse a dos niveles: por un lado, en el Modelo de Diálogo se establece una representación del diálogo a nivel de estados de la interfaz y, por otro lado, en el Modelo de Presentación Abstracto se establece una representación del diálogo a nivel de estados de componente o entre componentes individuales.

El estudio de los modelos de especificación de interfaces que se realiza a continuación se centra en la caracterización de su estructura, haciendo especial hincapié en como se realiza la definición de la apariencia visual de la interfaz (en el caso de que lo soporten).

3.3 Modelos de especificación de interfaces

En esta sección se presentan algunos de los modelos de especificación de interfaces de usuario más conocidos y utilizados. Para cada uno de ellos, se han identificado los pasos que contemplan en el diseño de interfaces. De forma resumida, pueden observarse en las tablas 3.1 y 3.2

3.3.1 Seeheim

El modelo Seeheim [CCCL93][Cou93] ([Pfa85],(1983)) sigue una aproximación lingüística. Divide los componentes de interfaz (interadores) en cuatro capas:

- Aplicación (Modelo Mental y Modelo de Tareas): engloba los conceptos específicos del dominio, es decir, la semántica
- Interfaz de Aplicación (Modelo de Aplicación): trata los aspectos léxicos. Contiene las estructuras de datos y las funciones asociadas a la parte mental del sistema que deben estar disponibles para la aplicación de interfaz. Además, es el encargado de analizar los datos antes de que éstos sean enviados al sistema.

	MM	MT	MA	MD
SEEHEIM	Aplicación	Aplicación	Interfaz Apl.	CtrolDiálog
MVC	Modelo	Modelo	Controlador	Controlador
ARCH	Dominio	Dominio	Adaptador	Diálogo
ALV	Abstracción	Abstracción	Link	Link
PAC	Abstracción	Abstracción	Control	Control
Pac-Amodeus	Cód. Func.	Cód. Func.	Adaptador	Controlador
JACOB			X	X
UAN				X
IRG			X	X
Monolítico	Aplicación	Aplicación	Aplicación	Aplicación
Cliente/Serv	Cliente	Cliente	Cliente	Cliente
CNUCE			X	X
YORK			X	X
ADV	ADT	ADT	ADT	ADV
IOG			X	X
ADC	Abstracción	Abstracción	Controlador	Controlador
UDM	X	X	X	
PUM		X		X
ICO	Comportam.	Comportam.	Comportam.	Comportam.

Tabla 3.1: Tabla de uso de las cuatro primeras etapas de diseño en los modelos

	MPA	MPC	MCA	MCC
SEEHEIM	Presentac.			
MVC	Vista/Contr.			
ARCH	Presentación	Toolkit Int.		
ALV	Vista			
PAC	Presentación			
Pac-Amodeus	Presentac.	Presentac.		
JACOB	X			
UAN	X			
IRG	X			
Monolítico	Aplicación			
Cliente/Serv	Servidor			
CNUCE	X			
YORK	X			
ADV	ADV			
IOG	X	X		X
ADC	Display		X	
UDM	X			
PUM	X			
ICO	Presentac.			

Tabla 3.2: Tabla de uso de las cuatro últimas etapas de diseño en los modelos

- Presentación (Modelo de Presentación Abstracto): especifica los objetos interactivos que componen la interfaz de usuario. Incluye los algoritmos que implementan los aspectos visuales de la interfaz y los que manejan los datos y comandos de entrada de usuario.
- Control de Diálogo (Modelo de Diálogo): Es una capa intermedia entre las de Presentación e Interfaz de Aplicación. Su función es controlar la cantidad de información que necesitan conocerse entre ambas capas. Representa la parte sintáctica del modelo.

Cabe destacar algunos refinamientos desarrollados bajo el modelo Seeheim.

Modelo-Vista-Controlador (MVC) Este modelo [CCCL93][HC95a] ([KP88],(1988)) trata con más atención la relación entre el modelo conceptual y el modelo mental del usuario. Considera que los objetos interactivos están formados por componentes externos (agentes cooperativos) que encapsulan su estado:

- Modelo (Modelo Mental y Modelo de Tareas): similar al componente Aplicación de Seeheim. Refleja la estructura del modelo conceptual; es la abstracción del objeto de interfaz.
- Vista (Modelo de Presentación Abstracto): representa la percepción que tiene el usuario de la abstracción. Depende del objeto Modelo. Corresponde a la parte del modelo abstracto de presentación relacionada con la salida. Las Vistas pueden organizarse bajo una jerarquía de modo que existan subvistas dentro de supervistas. La supervista tiene la capacidad de realizar transformaciones gráficas, manejo de ventanas y envío de información entre los diferentes niveles de la jerarquía.
- Controlador (Modelo de Diálogo y Modelo de Presentación Abstracto): trata los datos de entrada y comandos relacionados con el objeto de interfaz, y gestiona la información que intercambian el usuario y el componente Modelo. Establece cómo los componentes responden a los eventos, de modo que Vista se encarga de reflejar visualmente la respuesta a cada evento.

Su operativa habitual es: el usuario realiza una acción, el Controlador lo notifica a Modelo, Modelo realiza los cambios correspondientes; Vista puede interrogar a

Modelo para determinar si es necesario actualizar el aspecto visual. El Controlador se informa, directamente o a través de Vista, de los cambios de Modelo; de este modo se puede modificar dinámicamente la entrada de procesos al Controlador en base al estado del Modelo. El objeto Modelo suele encargarse de instanciar los objetos Vista y Controlador en el momento en que se inicializa; puede incluso provocar la creación de nuevos "subinteradores"(cajas de diálogo, botones, etc.) que dependen de él.

Los objetos interactivos pueden relacionarse de forma explícita o implícita. En el primer caso se realiza un paso de mensajes entre Modelos, mientras que en la relación implícita los interadores comparten Modelo (existe un único Modelo y varios pares Vista/Controlador) de modo que ya no es necesario el envío de mensajes. Un sistema con estructura de modelo compartido podría ser, por ejemplo, una ventana de texto con una barra de desplazamiento asociada.

La metáfora Modelo-Vista-Controlador se ha implementado en Smalltalk-80 [KP88] mediante la definición de tres superclases asociadas a cada uno de los componentes del modelo. La superclase Vista engloba el comportamiento genérico de las vistas en el sistema incluyendo:

- la interacción entre el Controlador y el Modelo
- la interacción entre supervistas y subvistas
- la transformación de coordenadas (traslación y escalado) entre una vista y sus supervistas y subvistas
- las acciones de colocación en el dispositivo de visualización. Se considera que los elementos incluidos en una Vista utilizan objetos de visualización predefinidos (Formularios, Líneas o instancias de clases gráficas), por lo que únicamente se almacena la transformación desde el sistema de coordenadas interno de la vista al del dispositivo de visualización (más concretamente, la transformación de la vista y de todas sus supervistas). Para cada vista es posible conocer el espacio libre en el que podría visualizar otras vistas sin dar lugar a solapamiento, el color y ancho de su borde, su color de fondo, y las dimensiones de los objetos de visualización dentro de su sistema de coordenadas. La colocación de las subvistas se lleva a cabo mediante la definición de rectángulos relativos.

Arch Bajo el modelo Arch [Cou93] ([Arc92],(1992))los interadores que componen la interfaz poseen los siguientes componentes:

- de Dominio Específico (Modelo Mental y Modelo de Tareas): corresponde a la capa Aplicación de Seeheim
- Adaptador de Dominio (Modelo de Aplicación): corresponde a la capa Interfaz de Aplicación de Seeheim. Funciona como capa intermedia entre la de Dominio Específico y la de Diálogo. Maneja *objetos de dominio*: entidades que el diseñador desea que sean manipulables por el usuario; empareja la representación mental del usuario con un concepto(s) de dominio particular(es).
- Diálogo (Modelo de Diálogo): corresponde al Control de Diálogo de Seeheim. Posee la responsabilidad de la secuencia de tareas. Además, debe mantener la consistencia entre los formalismos de Dominio Específico (las entidades que la capa de Dominio Específico intercambia con el Adaptador de Dominio) y los de la interfaz (información de Presentación). Debe realizar la transformación de datos y su mapeo. Por último, también es función de esta capa el mantener la consistencia entre las múltiples vistas de cada objeto conceptual (un objeto de dominio puede estar asociado a múltiples técnicas de presentación).
- el componente Presentación de Seeheim se divide en dos niveles de abstracción:
  - Presentación (Modelo de Presentación Abstracto): proporciona un conjunto de objetos (llamados *objetos de presentación*) independientes de las herramientas visuales finales. Actúa como mediador entre el componente Diálogo y el Toolkit de Interacción.
  - Adaptador al Toolkit de Interacción (Modelo de Presentación Concreto): implementa la interacción física con el usuario definiendo los denominados *objetos de presentación*. Estos objetos se mapean posteriormente al formalismo soportado por las herramientas (toolkits) de interacción.

Abstracción-Link-Vista (ALV) El modelo ALV [CCCL93], desarrollado por Hill ([Hil92], (1992)), considera los objetos interactivos compuestos por:

- Abstracción (Modelo Mental y Modelo de Tareas): es similar al componente Aplicación de Seeheim

- Vista (Modelo de Presentación Abstracto): es similar al componente Presentación de Seeheim. Existe un componente Vista por cada usuario

- Link (Modelo de Aplicación): corresponde al componente Interfaz de Aplicación de Seeheim. Su función es la de conectar las Vistas con el componente Abstracción, propagando eventos bidireccionalmente.

A diferencia del modelo Seeheim, la conexión entre los objetos de interfaz con la parte interna del sistema se realiza mediante restricciones: El componente Enlace contiene variables donde un objeto puede asociarse con funciones de otros objetos. Cuando el valor de la variable cambia se produce una llamada a la función asociada. ALV se orienta a sistemas multiusuario síncronos donde el código funcional está centralizado y es compartido.

3.3.2 Presentación-Abstracción-Control (PAC)

El modelo PAC [CCCL93][CCN97][HC95a], de Coutaz ([Cou87],(1987)), estructura una interfaz como una jerarquía de agentes (interadores). Cada interador está compuesto de:

- Abstracción (parte semántica, Modelo Mental y Modelo de Tareas): engloba los conceptos y las funciones del sistema.

- Presentación (parte léxica, Modelo de Presentación Abstracto): define la apariencia del interador y es responsable también de la comunicación entre la aplicación de interfaz y el código funcional, tanto desde el punto de vista de la entrada como de la salida. En el caso de la salida, proporciona únicamente especificaciones de rendering Semántico (de tipo Notificación y de Estado del Sistema). El componente de Presentación puede estar compuesto por otros objetos PAC.

- Control (modelos de aplicación y diálogo): almacena el estado local, permitiendo de este modo un diálogo multihilo. Se encarga de la comunicación entre interadores y de mantener la consistencia entre Abstracción y Presentación.

Variante: PAC-Amodeus El modelo PAC-Amodeus, desarrollado por Coutaz y Nigay ([CN91], (1991)), es un refinamiento del modelo Arch utilizando agentes PAC:

- Código Funcional (Modelo Mental y Modelo de Tareas): corresponde al Dominio Específico de Arch.

- Adaptador de Código Funcional (modelo de aplicación): es el componente Adaptador de Dominio de Arch.

- Controlador de Diálogo (Modelo de Diálogo): corresponde al componente Diálogo de Arch. Realiza las funciones propias del Modelo de Diálogo. Se implementa mediante un conjunto de agentes PAC cooperativos que se asocian a los diferentes estados de la interacción. Como los agentes tienen capacidad para mantener su propio estado, es posible suspender y recuperar posteriormente las tareas del usuario. Gracias a que los agentes permiten soportar secuencia de tareas, transformación de formalismos y mapeo de datos en múltiples niveles de abstracción, es posible definir el Controlador de Diálogo utilizando un paradigma de abstracción y refinamiento, refinando los agentes en otros más simples de forma recursiva. El estado interno de la interacción se define mediante el conjunto de todas las partes Abstracción de los diferentes agentes, el estado externo mediante las partes Presentación, y las funciones de mapeo entre los estados interno y externo mediante las partes Control.

- Presentación (modelos abstracto y concreto de presentación): engloba los dos niveles de abstracción del componente Presentación de Arch.

3.3.3 Lenguaje de especificación de Jacob

El lenguaje de Jacob ([Jac86],(1986)) permite la especificación de interfaces de manipulación directa mediante la definición de un conjunto de objetos interactivos, sus diálogos asociados (utilizando diagramas de transición de estados) y los eventos relacionados con las transiciones de estado. Para poder combinar los diálogos propios de cada objeto interactivo, y para poder incorporar la posibilidad de que los diálogos se "suspendan"y se "recuperen", se define un diagrama de estados inicial (el diálogo maestro) encargado de ejecutar los objetos interactivos como un conjunto de corrutinas, asignándoles los eventos de entrada y arbitrando el almacenamiento del estado suspendido.

De este modo, la especificación de los diálogos de manipulación directa queda dividida en tres niveles:

- nivel léxico (Modelo de Aplicación): descripción individual, a bajo nivel, de cada evento asociado con las transiciones de su diagrama de estados; son invocados desde la especificación a nivel sintáctico.
- nivel sintáctico (Modelo de Diálogo y Modelo de Presentación Abstracto): corresponde a las secuencias de eventos de entrada y salida representadas mediante los diagramas de estados. Cada evento se especifica, de forma individual y separada de la descripción del diálogo, mediante un conjunto de acciones de entrada, incluyendo detalles de aspecto, representación gráfica y manejo de dispositivos.
- nivel semántico (Modelo Mental y Modelo de Tareas): conjunto de procedimientos que implementan los requerimientos funcionales del sistema (pertenece al código funcional del sistema); son invocados desde la especificación a nivel sintáctico. No forman parte de la especificación de la interfaz.

El lenguaje de Jacob permite la descripción de *objetos contenedores* compuestos por varios objetos interactivos, teniendo cada objeto contenido su propia definición. Los objetos contenidos se instancian en el momento, y solo en ese momento, de la creación del objeto contenedor e igualmente son dibujados y destruidos. El uso de objetos contenedores permite agrupar secuencias de operaciones de más bajo nivel y ser procesadas por objetos de mayor nivel. Para ello se incluye la posibilidad de enviar eventos simplificados entre diagramas de sintaxis.

3.3.4 User Action Notation (UAN)

El modelo UAN [HD95], de Hartson ([HSH90],(1990)), es una notación desarrollada por los diseñadores de sistemas y está más relacionada con la complejidad de la interacción que con la complejidad del usuario. Su concepto central sigue siendo la tarea del usuario, pero ahora su elección es competencia del sistema, no del usuario. Soporta descomposición de tareas y refinamiento de cada tarea básica en la secuencia de acciones de usuario que pueden ejecutarla. Una interfaz se representa como una estructura casi-jerárquica de tareas asíncronas. Utiliza una notación tabular con 3 columnas: acciones de usuario, cambios de visualización y cambios de estado de la interfaz. Las tareas y/o acciones descritas se ejecutan de arriba a abajo y de izquierda a derecha. Cada tarea puede asociarse a una precondición.

UAN es una notación interesante porque permite describir aspectos propios de los

sistemas visuales que no son tratados adecuadamente en otras notaciones. Las técnicas basadas en gramáticas formales, por ejemplo, no permiten describir el feedback del sistema, y resulta muy complicado especificar tareas basadas en el arrastre de iconos ya que su semántica depende del lugar donde se suelte. UAN permite una descripción física de las acciones a realizar en el proceso de interacción.

UAN es adecuada para el análisis de tareas de usuario y de realimentación del sistema (Modelo de Diálogo). Sin embargo, se centra demasiado en la descripción de las acciones de usuario y no está bien adaptada para describir el estado del software. Su representación tabular hace difícil entender la relación entre los diferentes componentes de la interfaz de usuario (por lo tanto, no cumple todos los aspectos propios del modelo concreto de presentación). Por último, no proporciona ninguna manera de describir los cambios provocados a la interfaz por parte del sistema.

3.3.5 Variaciones de Diagramas de Transición de Estados: IRGs

Los modelos que utilizan diagramas de estados [FR82][Jac83][WS84] [SRH85] suelen representar los estados de la interfaz mediante nodos, y la secuencia de entradas del usuario mediante transiciones representadas como enlaces. Las salidas se especifican mediante anotaciones. Permiten mostrar el flujo de acciones y determinar el estado tras una acción. Se pueden utilizar para describir mecanismos complejos y resultan muy adecuados para la especificación de interfaces de usuario convencionales. Sin embargo, no resultan adecuados para descripciones de sistemas con diálogos concurrentes pues, en este caso, se produce una explosión combinatoria de estados. Los diagramas de estado de Harel ([Har88],(1988)) surgieron para resolver los problemas de explosión combinatoria. Añaden el concepto de metaestado que agrupa conjuntos de estados con transiciones comunes que son heredadas por todos los estados que engloba. Trabaja con 4 nuevos tipos de estados:

- secuenciales (metaestados XOR): contienen redes de transición secuencial
- paralelos (metaestados AND): contienen varias redes de transición. Cada red se ejecuta en paralelo funcionando como una corrutina
- estados historia: permiten devolver un metaestado a su estado anterior o retornar transiciones desde eventos como la invocación de ayuda. La diferencia entre

ambos tipos de estado historia es la forma de retornar cuando el último estado fue un metaestado (al estado inicial del metaestado o al estado almacenado).

En su forma original, no incorporan flujo de datos ni abstracción.

Los **Grafos de Representación de Interfaces (IRGs)de Rouff** ([Chr91], (1991)) extienden los diagramas de estado para representar el diálogo, incorporando:

- representación del componente de interfaz, físico o lógico, además del estado
- especificación del flujo de datos y de control, permitiendo restricciones
- herencia entre objetos de interfaz, flujo de datos, flujo de control y atributos
- especificación del feedback semántico entre el código funcional y la interfaz
- especificación de interacción entre widgets (no definición de nuevos widgets)

Añaden dos tipos de estados, adicionales a los existentes en los diagramas de estado: Objetos de Datos y Estados de Visualización.

Los IRGs permiten la definición de objetos de interfaz (modelo de aplicación) y la aplicación de aspectos propios de la orientación a objetos, como la definición de atributos o la herencia. El diálogo entre objetos (Modelo de Diálogo) se representa mediante grafos basados en los diagramas de estados tradicionales. Además, incorpora la posibilidad de representar los componentes de interfaz y su interacción (Modelo de Presentación Abstracto).

3.3.6 Monolítico

El modelo monolítico, ([CCCL93], (1992)), no separa las decisiones de diseño de interfaces. Desde el Modelo Mental hasta el de presentación se implementan en un único componente denominado Aplicación. Es un modelo poco recomendable para la especificación de sistemas. Tiene importantes deficiencias como se detallan en [Lar92].

3.3.7 Cliente-Servidor

Este modelo [CCCL93] ([Shi92], (1992)), engloba dos componentes: el Cliente y el Servidor. Se trata principalmente de un modelo de comunicación, no de implemen-

tación, por lo que no hay una división clara respecto a las decisiones de diseño de interfaces que se asignan a cada componente. Algunos gestores de interfaces imponen restricciones semánticas:

- El Cliente implementa las estructuras de datos y las funciones que soportan y manipulan los objetos conceptuales (modelos mental, de tareas, de aplicación y algunas decisiones de diálogo)
- El Servidor es responsable de presentar la información de la parte cliente al usuario, convertir los datos y comandos introducidos por el usuario y enviar los datos convertidos al cliente (Modelo de Presentación Abstracto y una parte del Modelo de Diálogo).

3.3.8 CNUCE

El modelo de CNUCE, [HD95] ([PF92], (1992)), se basa en la formalización de los componentes básicos (objetos interactivos) con los que se construyen los programas gráficos interactivos. La idea subyacente a este modelo es representar un componente activo que posee una apariencia gráfica y una medida correspondiente al valor actual (estado) del interador (modelos de aplicación y diálogo). El cambio de estado puede modificar la apariencia del interador. La noción de interador es independiente de los modelos de referencia particulares de los sistemas interactivos (modelo de presentación abstracta). Podemos sincronizar los distintos procesos mediante señales de control.

Los interadores están formados por cuatro componentes (Collection, Presentation, Measure y Abstraction) que intervienen en diferentes flujos:

- Flujo de información desde la aplicación hacia el usuario (rendering): Collection almacena una representación abstracta de la información manipulada y representada por el objeto interactivo, proporcionando una descripción de alto nivel de la apariencia gráfica del objeto. Cuando se hace un trigger, la información pasa a Presentation que posee las primitivas (por ejemplo, los comandos gráficos básicos) para que la información pueda ser perceptible por el usuario. CNUCE proporciona especificaciones de rendering de Diálogo (de Activación) y Semántico (de tipo Notificación).
- Flujo de información desde el usuario hacia la aplicación: Las entradas del

usuario se almacenan en Measure. Cuando se hace un trigger pasan a Abstraction donde se convierten a una representación abstracta más adecuada para su envío a la aplicación. Measure puede sustituirse también por una función en Collection que se aplique a los datos de entrada antes de pasar a Abstraction.

Para aplicar esta formalización a un sistema específico son necesarios dos pasos:

1. representar el sistema como un conjunto de objetos interactivos, y
2. describir los tipos de datos con los que trabajan los componentes de cada objeto interactivo.

3.3.9 York

En la Universidad de York se ha definido un marco formal para la definición de interadores [HD95] ([DH93], (1993)). Abstractamente, un interador (objeto interactivo) es un estado interno reflejado, mediante una relación de rendering, hacia una representación perceptible P (??). Los interadores están formados por estados, comandos, eventos y renderings (proporcionan especificaciones de rendering de Diálogo (de Activación) y Semántico (de tipo Notificación y de Estado del Sistema)). Su descripción es independiente de cualquier modelo de referencia para sistemas interactivos.

La noción de objeto interactivo como entidad independiente con un estado interno que puede modificarse mediante eventos producidos por su entorno, lo hace similar al concepto tradicional de objeto. Y como tal consta de:

- un espacio de estados σ
- un conjunto de estados iniciales i
- un alfabeto de eventos α, donde un evento es una abstracción de una acción o cambio en el sistema; constituyen la interfaz entre el entorno del objeto y su estado interno
- un conjunto de observaciones o trazas de eventos τ que capturan el comportamiento del objeto en el tiempo
- un conjunto de relaciones de transición de estados β

- una relación resultado R establecida entre trazas y estados (después de ejecutarse una determinada traza, un objeto puede estar en cualquier estado s$\in \sigma$ tal que $t^R \rightarrow$s)

Pero además, la descripción de un interador extiende la definición de objeto, introduciendo:

- el espacio de renderings P (representaciones perceptibles)

- una relación vista V entre trazas y renderings. Así, para la traza t y el rendering p, $t^V \rightarrow$p indica que después de la traza t se puede percibir p

- la relación de rendering ρ entre estados y renderings. Así, para el estado s y el rendering p, s ρ p indica que después de una determinada traza t, p es un posible rendering de s. En principio, se permite que dos trazas que construyan los mismos estados puedan producir, sin embargo, diferentes vistas. Una restricción más fuerte es forzar a que s ρ p signifique que cualquier traza que construya s pueda dar lugar exactamente a la vista p.

el estado y el rendering de un interador se ven como dos objetos enlazándose entre sí mediante eventos comunes. En este caso, el mantiene oculto, mientras que el estado del objeto de visualización) es visible.

Igual que el modelo CNUCE, YORK se basa en representar el sistema como un conjunto de objetos interactivos (interadores). Extiende la definición de los objetos, como un conjunto de estados y transiciones entre los mismos (Modelo de Aplicación y Modelo de Diálogo), centrándose en la representación de su estado actual, los eventos para su manipulación y sus posibles visualizaciones. La noción de interador es independiente de modelos de referencia particulares para sistemas interactivos o gráficos (modelo de presentación abstracta). El modelo permite formular y verificar propiedades generales de interacción, pero no pretende ser un lenguaje de especificación de sistemas interactivos. Sin embargo, la noción de interador proporciona una base útil para la descripción y el razonamiento del comportamiento de sistemas interactivos. Permiten capturar principios generales de interacción.

3.3.10 Vista de Datos Abstracta (ADV)

Este modelo, desarrollado por Cowan et al ([CILS93], 1993), proporciona una clara separación entre la parte de la aplicación y la interactiva. Un sistema típico basado en este modelo consta de [CCCL93][CCL93]:

- un conjunto de Tipos Abstractos de Datos (ADTs) que gestionan las estructuras de datos y el estado de la aplicación (modelos mental, de tareas y de aplicación). Son totalmente independientes de la interfaz interactiva.
- un conjunto de Vistas Abstractas de Datos (ADVs) relacionadas con el comportamiento interactivo engloban vistas y eventos (modelo de diálogo y abstracto de presentación). Proporcionan las funciones de entrada y salida y controlan su ADT asociado.
- un mapeo entre ambos conjuntos. El mapeo asocia las vistas de las variables en ADV con el estado de las variables en ADT de modo que cualquier modificación se verá reflejada en ambos elementos (modelo de diálogo).

Una instancia ADV se asocia con exactamente una instancia ADT, pero una instancia ADT puede asociarse con varias instancias ADV. A diferencia de los modelos anteriores, el control se encuentra en la parte de la interfaz (el nivel más alto), no en el sistema interno (el nivel más bajo), lo cual resulta muy adecuado en sistemas altamente interactivos. El uso de múltiples ADVs, uno por cada vista independiente, simplifica el flujo de control: cada ADV debe responder a un número relativamente pequeño de eventos de usuario y manipula un único ADT.

3.3.11 Grafos de Objetos de Interacción (IOGs)

Los Sistemas de Gestión de Interfaces de Usuario (UIMS) actuales asumen la existencia previa de un conjunto de primitivas (objetos de interacción o widgets) de modo que únicamente es posible manipular su presentación. Los Grafos de Objetos de Interacción (IOG), ideados por Carr [Car94] (1994), son una aproximación al problema de construcción de nuevos widgets permitiendo definir no solo su comportamiento sino también su posicionamiento en la interfaz y sus relaciones espaciales. Puede suponer la base para extender los UIMSs de modo que admitan el diseño y prototipado de nuevos objetos de interacción.

Los IOGs se basan en los Grafos de Representación de Interfaces (IRGs) permitiendo, además, la especificación de objetos de interacción de bajo nivel. Proporcionan especificaciones de rendering de Diálogo (de tipo Estado del Diálogo y Activación) y Semántico (de tipo Notificación, Estado del Sistema e Informes de Error). A diferencia de los IRGs, los estados Objetos de Datos representan únicamente el almacenamiento de datos, mientras que Estados de Visualización son estados de control que provocan un cambio en la apariencia del widget. Respecto a los arcos de transición, aparte de los arcos de control posee otros tres tipos:

- arcos de eventos: permiten definir mensajes útiles para el modelo de especificación subyacente (eventos definidos por el usuario). De este modo se pueden asociar otro tipo de condiciones a las transiciones. Este tipo de eventos está presente también en el lenguaje de Jacob.
- arcos de datos: enlazan un estado fuente con un Objeto de Datos o una terminación. Si el origen es un Objeto de Datos y el destino una terminación, indica que los datos son exportables. Si el origen es un Estado de Visualización indica modificación de datos si se cumple la condición del arco. Los arcos cuyo destino es un Objeto de Datos representan actualización de datos. Por último, un arco sin fuente indica actualización externa del objeto.
- arcos de restricciones: permite especificar unos atributos en función de otros (por ejemplo, limitar un icono a estar contenido en una ventana). Solo admiten interacciones entre dos Objetos de Datos.

Para describir las transiciones entre estados se necesita un modelo abstracto de la interfaz. IOG abstrae la interfaz en los siguientes objetos:

- Booleanos, números (enteros y reales) y strings: cualquiera de ellos se puede convertir a una representación textual y esta, a su vez, a un icono.
- Puntos: pares ordenados (x,y). Tienen los operadores algebraicos habituales.
- Regiones: conjuntos de puntos visuales. Se definen de forma relativa al extremo superior izquierdo de la región. Un operador tamaño indica el tamaño del rectángulo más pequeño que cubre la región. No hay una operación de dibujado para las regiones.

- Iconos: regiones con imágenes (es decir, con puntos asociados a un número de color que se muestran en el display)
- Puerto de vista: región con una función asociada que la convierte a una representación gráfica, a un sistema de coordenadas real, y la proyecta en el display.
- Ventana: agrupa objetos mapeados. Permite establecer niveles (una ventana con menor nivel oscurece una ventana solapada con mayor nivel)
- Entradas de Usuario: se mapean a eventos, números, puntos y variables booleanas.

IOG es un método que permite especificar y comunicar diseños de nuevos objetos de interacción. Frente a otros métodos, como los diagramas de estado o el modelo UAN, muestra de forma más clara el aspecto visual de los objetos y sus cambios dinámicos de apariencia por cambios en los datos del sistema asociados (modelos de presentación y de layout). Desafortunadamente, tiene algunos problemas. Por un lado, el nivel de detalle necesario para especificar un objeto interactivo aumenta considerablemente la especificación. Una primera solución a este problema es especificar únicamente los atributos que afecten al comportamiento dinámico del IOG y definir los restantes en una lista. Otro problema es la necesidad de codificación. Sería deseable disponer de una herramienta que permitiese editar un IOG y ejecutarlo directamente.

3.3.12 Abstracción-Display-Controlador (ADC)

ADC, de Markopoulos ([Mar95], (1995)), es un modelo formal para la construcción de especificaciones de interfaces mediante la composición de interadores. Realiza una descripción separada de los datos que alteran el comportamiento del interador frente a las restricciones temporales impuestas por el propio comportamiento. Utiliza diferentes representaciones para los datos de "visualización"y de "resultados", y los relaciona mediante una herramienta algebraica de proceso, consiguiendo así expresar las propiedades de la interacción relacionadas con la usabilidad. El modelo ADC distingue dos aspectos en la descripción del interador:

- comportamiento de datos: se modela mediante la Unidad de Abstracción y Display (ADU) cuya función es la comunicación de datos entre los diferentes niveles

de abstracción. Mantiene una representación del estado del interador (denominada Abstracción) y de su salida (el Display). Abstracción se corresponde al componente Modelo de MVC y al Abstracción de PAC; Display es similar al componente Vista de MVC.

- información de control: gestionada mediante el componente Controlador(C). Se encarga de la ordenación temporal del comportamiento del interador.

Los datos enviados por el usuario o la aplicación son entrada al ADU que actualiza su información de estado. Luego emitirá el estado de display hacia el usuario o una interpretación del estado de abstracción, el resultado, al lado de la aplicación. El ADU no impone restricciones en el comportamiento temporal del interador: las secuencias de eventos válidas son definidas por el Controlador como restricciones en el comportamiento del ADU.

El componente Abstracción engloba los conceptos y las funciones del sistema. Corresponde a los modelos mental y de tareas. El componente Display corresponde a una parte del modelo abstracto de presentación, en concreto a la relacionada con la salida. Por último, el Controlador trata los datos de entrada y comandos relacionados con el objeto de interfaz, y gestiona la información que se intercambia el usuario y el sistema. Abarca, por tanto, los modelos de diálogo y una parte del modelo de presentación.

Aunque este modelo es similar a los interadores de PAC, hay una importante diferencia: El Controlador de PAC gestiona toda la comunicación entre los componentes de presentación y abstracción, y traslada los datos entre los dos formalismos. Esta comunicación se oculta en el Controlador de ADC que simplemente impone restricciones externas de diálogo en sus operaciones.

3.3.13 Método de Diseño Unificado (UDM)

El propósito principal de la información recogida durante el proceso de diseño es facilitar la implementación de los objetivos y/o la evaluación de la usabilidad. En el contexto de la interacción adaptativa, donde las aplicaciones se adaptan por sí mismas a los requerimientos del usuario final, es preciso diseñar diferentes patrones que reflejen los requerimientos propios de cada usuario. El Método de Diseño Unificado (UDM), de Savidis et al [SPAS97](1997), permite unificar la representación del diseño del diálogo

de las interfaces adaptativas. La estructura de tareas polimórficas, en combinación con la representación explícita de los atributos de usuario y la expresión de la lógica para la selección de diálogos alternativos, se mapea directamente a un modelo de arquitectura basada en componentes. Se basa en la noción de descomposición polimórfica de tareas: cada tarea puede descomponerse en una serie de subjerarquías alternativas. El proceso de diseño realiza una exhaustiva descomposición jerárquica de las tareas de usuario, comenzando desde el nivel más abstracto hasta el nivel físico de interacción. El método de descomposición se basa en 3 elementos:

- organización jerárquica: similar al análisis de tareas habitual, permite la descomposición incremental de tareas de usuario en acciones de más bajo nivel
- polimorfismo: proporciona la capacidad para diferenciar el diseño, en los diferentes niveles de jerarquía, en función de los requerimientos propios del usuario
- operadores de tareas: expresan el flujo de control del diálogo

Durante el proceso de descomposición de tareas, algunas subtareas pueden relacionarse directamente con entradas del usuario a través de objetos interactivos. En este caso, es necesario definir clases abstractas que incluyan clases de objetos físicos alternativos que puedan seleccionarse en función de los diferentes requerimientos de usuario, diseño y dominio. UDM define un modelo basado en reglas que permite filtrar las decisiones de diseño e identificar los "puntos"donde deben emplearse objetos interactivos abstractos.

Los objetos interactivos se pueden agrupar en 3 categorías:

- léxica: el objeto se emplea para presentación (apariencia).
- sintáctica (de diálogo): el objeto tiene una función específica en el diseño de la secuencia de diálogo.
- semántica: en este caso, el objeto interactivo está asociado con un objeto conceptual.

UDM se basa en la descripción de las tareas conceptuales (modelo de tareas) y su descomposición en subjerarquías alternativas atendiendo a la posibilidad de la existencia de diferentes tipos de usuarios. Considera también la necesidad de definir objetos de

interfaz (Modelo de Aplicación) que permitan asociar las tareas con la interacción directa del usuario. Asocia los objetos interactivos con objetos conceptuales (definidos en el modelo mental), los utiliza en el diseño de la secuencia de diálogos (Modelo de Diálogo), o los define para presentación (este tipo de objetos se encuadra dentro de nuestro modelo abstracto de presentación, pues no trata los aspectos propios de su implementación física).

3.3.14 Modelos de Usuario Programables (PUM)

PUM, de Butterworth et al [BB97](1997), es una herramienta para la evaluación de interfaces de usuario en base a propuestas de las mismas, sin esperar a la implementación final. El analista propone un diseño de interfaz, el conocimiento requerido y el comportamiento deseado (éste se representa mediante un proceso planeado del diseñador (DIP) que engloba las secuencias válidas de transiciones entre estados para llegar a una meta). Si el modelo no produce el comportamiento deseado en algún caso, el analista debe rediseñar la interfaz o indicar la necesidad de añadir más conocimiento previo. Una vez que el analista está de acuerdo con el modelo de interfaz, se refina hasta obtener la interfaz final. A diferencia de otros modelos de usuario, PUM considera el conocimiento que el usuario necesita para interactuar con los dispositivos. El nivel de conocimiento permite que éste se capture de forma independiente a las estructuras cognitivas y a los mecanismos que lo almacenan (Modelo de Tareas). Lo importante es extraer el efecto que el conocimiento tiene en el comportamiento. De este modo, dos piezas de conocimiento simbólicamente diferentes, pero que dan lugar al mismo comportamiento, desde el nivel de conocimiento se consideran la misma pieza. El diseñador debe justificar de donde proviene cada clase de conocimiento que necesita el usuario. De este modo, si un elemento de conocimiento es imprescindible y no es razonable asumir que el usuario tiene ese conocimiento a priori, entonces la interfaz debe modificarse para que indique al usuario ese conocimiento. El conocimiento del usuario se representa como un conjunto de operaciones que puede realizar. Dichas operaciones no tienen por qué ser las mismas que las acciones de los dispositivos las operaciones expresan lo que el usuario conoce acerca del efecto de realizar un conjunto de acciones sobre dispositivos (Modelo de Diálogo). PUM parte de un modelo muy abstracto de los dispositivos (Modelo de Presentación Abstracto) y representa como sus recursos son compartidos entre él y el usuario.

El modelo se basa en el conocimiento que el usuario debe tener para tratar con la interfaz. Considera la necesidad de modelar el conocimiento, la arquitectura cognitiva, el dispositivo, la interacción y el comportamiento. Trabaja a diferentes niveles de abstracción:

- parte del modelo del dispositivo (modelo de presentación abstracto), describiendo el espacio de estados, las acciones válidas para cambiar entre estados y el comportamiento obtenido.
- describe el espacio cognitivo (Modelo de Tareas)
- define el espacio de estados de la interfaz y las acciones de usuario que describen bajo qué circunstancias el usuario invoca las acciones de los dispositivos (modelo de diálogo).

3.3.15 ICO

El formalismo ICO, de Bastide et al [BPLM98](1998), se basa en conceptos tomados de la aproximación OO (instanciación dinámica, clasificación, encapsulación, relación cliente/servidor, herencia) para describir la parte estructural (estática) del sistema, y emplea redes de Petri de alto nivel para describir los aspectos de comportamiento (dinámicos). Un objeto interactivo (interador) en ICO está compuesto por cuatro componentes:

- Comportamiento: muestra cómo reacciona el objeto ante un estímulo externo en función de su estado interno. Se describe mediante una red de Petri.
- Servicios: que el objeto ofrece al usuario o a otros objetos
- Estado
- Presentación: representa el aspecto externo del objeto. Se estructura como un conjunto de widgets organizados en un conjunto de ventanas.

La arquitectura ICO puede compararse fácilmente con la del modelo Seeheim. El componente Comportamiento juega un papel similar al componente Control de Diálogo de Seeheim: establece la correspondencia entre la parte de interfaz y la de aplicación. Las capas Aplicación e Interfaz de Aplicación de Seeheim se modelizan mediante los

	MD	MPA	MPC	MCA	MCC
Propuesta	X	X		X	

Tabla 3.3: Fases de diseño contempladas en la propuesta

eventos asociados a la red de Petri. Por lo tanto, Comportamiento afecta a los modelos mental, de tareas, de aplicación y de diálogo. La capa de Presentación de Seeheim se construye en ICO mediante un conjunto de interadores (widgets). Corresponde al Modelo de Presentación Abstracto.

3.4 Introducción de la propuesta

Para concretar la propuesta del presente trabajo (figura 3.1) se expondrá a continuación las fases de diseño que ésta abarca (Tabla 3.3).

- Modelo de Presentación Abstracto. Representa de forma abstracta la apariencia y representa de forma abstracta el diálogo entre componentes individuales.
- Modelo abstracto de composición. Representa de forma abstracta la colocación espacial topológica de los componentes individuales dentro de la interfaz de usuario final.
- Modelo abstracto de diálogo. Representa de forma abstracta el diálogo existente entre los componentes de la interfaz. A partir de las abstracciones anteriores y esta puede extraerse el modelo abstracto de diálogo entre estados de la interfaz.

Para cada una de las fases de diseño de la propuesta se expondrá un sistema de representación de la misma, a partir de la cual se podrá extraer más información acerca de la funcionalidad y estructura de la interfaz representada y permitirá la evaluación de características de la interfaz de forma cuantitativa.

3.5 Conclusiones del estudio de los modelos

Dada la importancia de la componente visual de la interfaz se han estudiado los modelos más importantes para la especificación o representación de la interfaz de usuario.

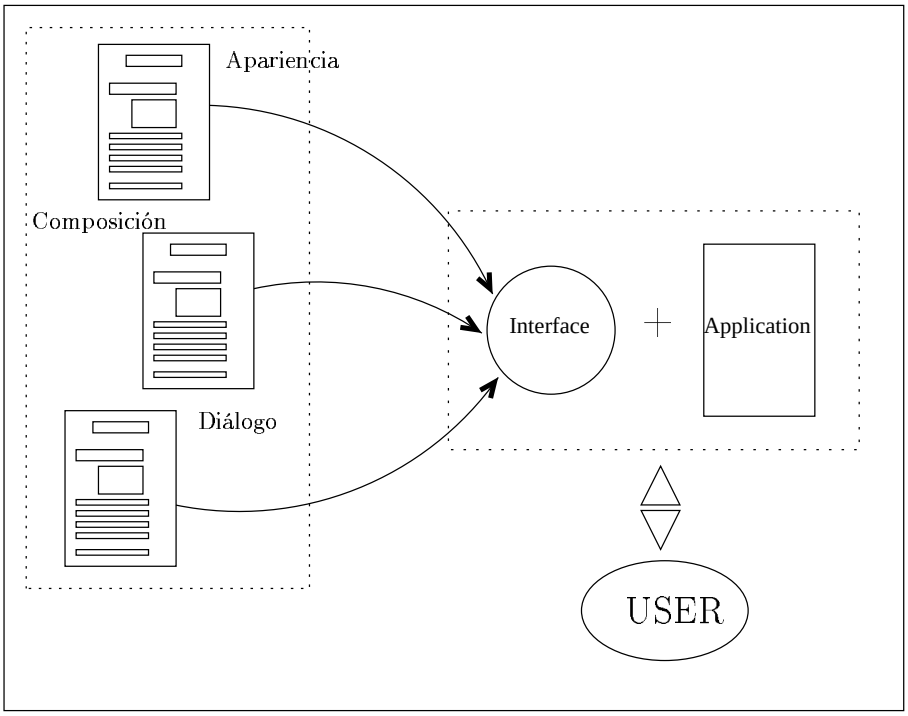

Figura 3.1: Esquema de la propuesta

Una vez examinados podemos ver que existe muy poca literatura acerca de la representación de la componente visual de la interfaz. En algunos modelos se contempla la presentación abstracta de la interfaz pero no la concreta, con lo cual esta parte está supeditada a la opinión de un implementador, y por lo tanto puede no reflejar todas las características de la interfaz identificadas por el modelo.

La mayoría de los modelos que incorporan representación abstracta de la interfaz lo hacen a nivel de comportamiento y solo algunos realizan una abstracción de la representación concreta de la interfaz (a nivel de visualización). Prácticamente todos los que incorporan representación concreta de la interfaz es porque incorporan herramientas de generación automática de código de interfaz.

Es muy importante resaltar que los modelos que realizan una representación abstracta o concreta lo hacen a nivel de componente y solo uno de todos ellos hace referencia a una representación de la colocación de los componentes dentro de la misma. Esta falta de representación de la composición de los componentes para generar la estructura final de la interfaz, junto con la necesidad de contemplar el comportamiento dinámico actual de los componentes, muestran una deficiencia en los modelos actuales.

3.6 Herramientas

Vamos a considerar dentro de las herramientas dos apartados diferentes. Por un lado, las herramientas y técnicas comunes de prototipado, por ser la forma más habitual de desarrollo de la interfaz de usuario. Por otro lado, trataremos en profundidad un tipo particular de estas herramientas de prototipado debido al especial uso que hacen de los modelos teóricos de definición de interfaces de usuario.

3.6.1 Herramientas de prototipado

Se realiza una clasificación de las herramientas y técnicas para el prototipado basándose en dos criterios: por un lado, los tipos de información necesarios para un correcto proceso de prototipado, y por otro en función del conjunto de requerimientos que las herramientas y técnicas de prototipado deberían satisfacer a nivel de diseño de la interfaz de usuario. En el presente trabajo, nos centraremos en la identificación de la relación existente entre la información que se considera necesaria para el proceso de prototipado asociándola con la información obtenida en los pasos de diseño de interfaces de usuario. Se identifican ocho tipos de información que es necesario poseer sobre el sistema que se va a construir para poder realizar un prototipo correcto de la interfaz de usuario. Esta información abarca desde el análisis de las tareas que los usuarios esperan poder realizar con el sistema hasta descripciones detalladas de la "apariencia" (look) y el "comportamiento" (feel) del sistema.

Especificación de Tareas La especificación de tareas se necesita para que los desarrolladores puedan entender qué servicios deben proporcionar en el sistema y cómo deben enviarlos a los usuarios finales para ayudarles a realizar sus tareas de forma más efectiva. Los prototipos ayudan a los desarrolladores a entender mejor las tareas que los usuarios necesitan, dado que pueden ver a los usuarios en acción con el sistema y obtener realimentación acerca de su efectividad.

Funcionalidad del Sistema Esta información especifica los requerimientos de los módulos software que la interfaz llama para colocar los datos en el dispositivo de salida, y para modificar los datos en respuesta a las peticiones del usuario. Frecuentemente, diseños de interfaz diferentes imponen diferentes requerimientos en la funcionalidad del sistema. Son especialmente útiles las herramientas

que pueden construir prototipos de la interfaz sin requerir que la funcionalidad del sistema sea construida.

Funcionalidad de la Interfaz Es una especificación de la información y estado del sistema que se necesita presentar al usuario, y los comandos que deben estar disponibles para el mismo. Esta especificación captura el "contenido"de la interfaz, abstrayendo la misma de detalles de estilo como fuentes, color, etc.

Composición de Pantalla y Comportamiento Especifica cómo se ve y cómo se comporta la interfaz. Esta información es muy importante dado que define lo que los usuarios van a ver y hacer. El prototipado es muy útil para adquirir esta información, porque el emphlook and feel permite finalizar las discusiones entre los miembros del equipo de diseño y existen muchas posibilidades que necesitan ser exploradas.

Diseño Racional Especificación de las razones que llevan a las diferentes decisiones de diseño. Puede utilizarse esta información para lograr consistencia en la interfaz, para guiar ampliaciones de la interfaz desde una versión previa del sistema o para revisar y justificar diseños. Además, cuando el prototipo solo incluye un subconjunto de las funcionalidades del sistema puede usarse para extrapolar desde el prototipo al sistema completo.

Realimentación desde el Usuario Es un log de realimentación acerca de la interfaz obtenida desde varias fuentes como los usuarios finales, gestores o revisores del diseño. La realimentación puede tomar múltiples formas: comentarios de los usuarios expresados mientras interactúan con el prototipo, respuestas a cuestionarios, segmentos de vídeo e historias completas de interacción con el prototipo.

Tiempos de Respuesta Corresponde con una especificación de los tiempos de respuesta requeridos en diferentes situaciones. Los prototipos permiten a los desarrolladores ver a los usuarios en acción y entender cuáles son los tiempos de respuesta adecuados para un uso efectivo del sistema.

Código Reutilizable Como consecuencia o resultado de construir un prototipo puede aparecer código reutilizable que puede ser usado en la implementación del sistema real. La construcción de un prototipo puede resultar costosa, y el coste es difícil de justificar cuando la implementación del prototipo no puede ser reutilizada en la implementación del sistema real.

Ahora vamos a relacionar los pasos del diseño de interfaces de usuario con la información que se indica como adecuada y necesaria para un correcto desarrollo de prototipado, teniendo en cuenta que de las informaciones anteriores no se incluíran las correspondientes a evaluaciones realizadas sobre el interfaz, que consideramos posteriores al proceso de diseño, ni las correspondientes a documentación. Entre la información de evaluación está la Realimentación desde el Usuario, los Tiempos de Respuesta y el Código Reutilizable, y entre las correspondientes a documentación y justificación de diseño se encuentra el Diseño Racional.

La información de Especificación de Tareas corresponde con Modelo de Tareas y tiene gran relación con el Modelo Mental del usuario. En el proceso de prototipado se permite al desarrollador entender la forma de pensar y trabajar del usuario y las tareas que éste pretende realizar.

La información correspondiente a la Funcionalidad del Sistema se relaciona con el Modelo de Aplicación, dado que se realiza la especificación de los módulos software relevantes para el prototipado y relacionados con la interfaz de usuario, y se definen las bases de funcionamiento de los módulos software del sistema. Por lo tanto, determina en gran manera la funcionalidad del sistema y, en consecuencia, la interfaz de usuario del mismo.

La Funcionalidad de la Interfaz proporciona información correspondiente al Modelo de Presentación Abstracto. Describe la información que debe estar disponible para el usuario en la interfaz, junto con indicaciones de cómo debe presentarse y que puede hacer el usuario con la misma, de forma independiente a su componente visual. En este sentido tiene mucha relación con los Objetos de Interfaz Abstracto (AIO).

Mediante la Composición de Pantalla y Comportamiento se proporciona información acerca de cuál es el aspecto visual de la interfaz, y consecuentemente de cada componente individual de la misma, y el comportamiento que tiene asociado. Contempla, por lo tanto, descripciones del Modelo de Diálogo y del Modelo de Presentación Concreto, dando para el primero la descripción de funcionalidad de la interfaz para el usuario y el comportamiento de la misma, y para el segundo el aspecto visual final que el sistema va a poseer.

Existen seis grandes grupos de herramientas a la hora de realizar prototipado:

Información prototipado	Pasos en el diseño de IU
Especificación de Tareas	Modelo Mental / Modelo de Tareas
Funcionalidad de Sistema	Modelo de Aplicación
Funcionalidad de la Interfaz	Modelo de Presentación Abstracto
Composición de Pantalla y Comportamiento	Modelo de Presentación Concreto y Modelo de Diálogo

Tabla 3.4: Relación entre la información de prototipado y los pasos del diseño de IU

Papel y Lápiz El Papel y Lápiz son las herramientas más populares utilizadas para describir diseños de interfaz. Bajo esta categoría se pueden incluir versiones electrónicas de estas herramientas como editores gráficos, de dibujo y de texto.

El papel y lápiz tiene grandes ventajas:

- Fácil de usar : mucha gente dibuja cajas como botones, menús y figuras (scribbles) representando objetos en un dominio de aplicación.
- Permiten un control extensivo sobre los detalles del diseño. No se limita a la habilidad de la persona a la hora de dibujar, dado que un diseñador puede dibujar una figura (scribbles) e indicar a los demás que significa.
- Enfatizan el trabajo en grupo, puesto que varias personas pueden estar diseñando al mismo tiempo, sobre todo cuando se realizan borradores sobre encerados o paneles grandes de papel.
- Son muy útiles ya que permiten capturar diferentes tipos de información, pues no es necesario establecer un dibujo sino expresarlos mediante anotaciones textuales.

Su principal desventaja está en la debilidad que posee para capturar comportamiento y que los prototipos de la interfaz no son ejecutables. El comportamiento, en la mayoría de los casos, está representado mediante dos dibujos de la interfaz, uno antes y otro después de la acción, junto con una anotación de cúal es la acción.

Facade Tools Las Facade Tools son esencialmente editores de dibujo con la propiedad de poder especificar comportamiento del usuario. Se han denominado así porque permiten a los desarrolladores construir pantallas que parecen y se comportan como

las pantallas de la aplicación real, exceptuando que no existe una aplicación detrás de las mismas.

Las herramientas de esta categoría se diferencian mucho en la calidad de los dibujos y en la sofisticación del comportamiento del usuario que puede ser especificado.

Herramientas que pertenecen a esta categoría son, por ejemplo, Astound [Inc93], Hypercard [Cor92], y MacroMind Director [Mac90].

Las Facade Tools poseen la mayoría de los beneficios del papel y lápiz, al tiempo que añaden la capacidad para definir comportamiento y ejecutar los prototipos.

Su principal desventaja está en que no producen código reutilizable que pueda ser usado para construir la aplicación real, de forma que el esfuerzo realizado en la construcción de la interfaz se pierde. Además, dado que las herramientas de prototipado e implementación son diferentes, deberían ser utilizados por diferentes equipos de diseño y habitualmente muchas de las cosas aprendidas durante el proceso de prototipado se pierden y no se utilizan en el proceso de implementación.

Interface Builders Las Interface Builders son herramientas de construcción de interfaces más que herramientas de prototipado. Su principal aportación es que, como las Facade Tools, proporcionan a los desarrolladores una interfaz similar a una aplicación de dibujo para especificar la interfaz pero, al contrario que las Facade Tools, generan código ejecutable que puede ser unido a la aplicación para producir una implementación comercial. Entre las herramientas de Interface Builders encontramos NeXT Step, Prototyper para Macintosh, WindowsMaker para Microsoft Windows y UIMX para X Windows y Motif. En muchos casos se venden como herramientas de prototipado porque cumplen muchas de las exigencias de este tipo de herramientas de prototipado. Son fáciles de usar por la amigabilidad de su entorno, permiten un feedback rápido de las modificaciones realizadas sobre la interfaz, así como un control exhaustivo sobre los diseños (dado que los diseñadores pueden cambiar fácilmente todas las propiedades de los componentes de la interfaz) y los prototipos son ejecutables.

Entre sus inconvenientes encontramos que solo pueden ser usados para construir *las porciones estáticas de la interfaz* (como, por ejemplo, menús y cuadros de diálogo que controlen la aplicación). No pueden utilizarse para especificar ventanas de aplicaciones donde la información de la aplicación se muestra de forma gráfica. Otro inconveniente

	MM	MT	MA	MPA	MPC	MD
Papel y Lápiz	X	X	X	X	X	X
Facade Tools				Implícito	X	X
Interface Builders				Implícito	Sólo estático	X
Dominio Específico	X	X	X	X	X	X

Tabla 3.5: Pasos del diseño de IU contemplados por las herramientas de prototipado

está en lo difícil que es aislar la interfaz del resto de la aplicación. Y, por último, los desarrolladores trabajan con componentes concretos para especificar las interfaces de usuario, forzándolos a aceptar características especiales de la interfaz antes de que estén preparados para ello.

Herramientas de Dominio Específico Son herramientas para construir tipos especiales de aplicaciones (por ejemplo, aplicaciones de bases de datos) o aplicaciones con estilos de interfaces específicos. Se examinan aquí porque el esfuerzo necesario para construir aplicaciones es frecuentemente menor que el necesario para construir prototipos con otras de las herramientas aquí mencionadas.

Un tipo representativo de las Herramientas de dominio Específico son las herramientas de cuarta generación (4GL). Son lenguajes de programación específicos para construir aplicaciones de bases de datos. El lenguaje proporciona utilidades para definir esquemas de bases de datos, interrogar y actualizar la base de datos y definir formatos que permitan a los usuarios realizar estas operaciones. Aunque las 4GL son herramientas de implementación, satisfacen muchos de los requerimientos indicados para las herramientas de prototipado. Proporcionan rápida respuesta a los cambios, control exhaustivo sobre los detalles de diseño, restricciones en el dominio de la bases de datos y, por supuesto, son ejecutables. Entre sus limitaciones están que suelen ser muy lentas para aplicaciones de gran escala, comparadas con otras herramientas de prototipado son difíciles de usar y requieren entrenamiento, aunque son usadas fácilmente por gente sin formación específica en informática.

De lo visto hasta el momento sobre herramientas de prototipado podemos concluir lo siguiente:

- Las herramientas de prototipado abarcan un abanico muy amplio de técnicas y tecnologías, desde una técnica absolutamente descriptiva y sin utilización de tec-

nología, como es la de Papel y Lápiz hasta una técnica absolutamente específica asociada al dominio sobre el que se utiliza (4GL).

- Aunque algunas contemplan hasta el Modelo de Presentación Concreto, están siempre sujetas a un entorno de implementación, y no proporcionan una definición abstracta del Modelo de Presentación Concreto, con lo cual no es factible su exportación a otro entorno de implementación.
- Ninguna de las herramientas incluyen un Modelo de Composición que indique la colocación de los componentes, siendo éste definido por el diseñador en el momento del prototipado, basándose únicamente en el Modelo de Presentación Concreto.

3.6.2 Herramientas basadas en Modelos

Otra de las posibilidades existentes dentro de las herramientas para el diseño de las interfaces de usuario son las MB-UIDE (Model-Based User Interface Development Environments). Estas se utilizan como base para la contrucción de Modelos Declarativos (UIM: Declarative User Interface Model) como parte del diseño de una interfaz de usuario.

Según [Sil00], los modelos declarativos de interfaces de usuario tienen tres grandes ventajas:

- Permiten una descripción más abstracta de las interfaces de usuario respecto a otras herramientas de desarrollo de interfaces de usuario.
- Facilitan la creación de métodos para diseñar e implementar las interfaces de usuario de una forma sistemática, dado que ofrecen capacidades como:
  - Modelar interfaces de usuario usando diferentes niveles de abstracción.
  - Refinar incrementalmente los modelos.
  - Reutilizar las especificaciones de las interfaces de usuario.
- Proveen de las infraestructuras necesarias para automatizar tareas relacionadas con los procesos de diseño e implementación de las interfaces de usuario.

La principal desventaja de los modelos declarativos de interfaz de usuario es la complejidad de los modelos y sus notaciones. Sin embargo, esta desventaja podría suavizarse

definiendo un entorno apropiado que proporcione utilidades como editores gráficos y asistentes para ayudar en el diseño.

Por ahora existen problemas no resueltos en los MB-UIDE, como son:

- Es díficil demostrar que los UIM describen aspectos relevantes de las interfaces de usuario necesarios para generar interfaces de usuario ejecutables. De hecho, hay pocos ejemplos de interfaces de usuario ejecutables generadas a partir de UIM declarativos.
- Múltiples publicaciones indican cómo integrar las interfaces de usuario con sus aplicaciones correspondientes, pero no son completamente aplicables a interfaces de usuario generadas por MB-UIDE.
- No existe consenso en cuanto a cuál es el conjunto de modelos que son más adecuados para describir las interfaces de usuario. De hecho, ni siquiera existe consenso en qué aspectos de las interfaces de usuario deberían ser modeladas.

La primera generación de MB-UIDE ejecutaba interfaces de usuario representadas de una manera declarativa. Proporcionaban una forma propia de ejecutar una interfaz de usuario desde un UIM declarativo. Ejemplos de esta generación son : COUSIN, HUMANOID, MIKE, UIDE o UofA. No proveían de niveles altos de abstracción de la interfaz de usuario, y se concentraban exclusivamente en la composición y la construcción de widgets.

La siguiente generación provee de niveles más altos para describir las interfaces de usuario. Es el caso de: ADEPT, AME, DIANE+, FUSE, MASTERMIND, MECANO, MOBI-D, TADEUS, Teallach o TRIDENT. Permiten especificar, generar y ejecutar interfaces de usuario. Algunas de estas también utilizan herramientas CASE y notaciones como UML.

[Sil00] realiza una clasificación sobre la utilización de los modelos por parte de once de las herramientas MB-UIDE actuales más representativas. Además se añaden al estudio anterior las herramientas desarrolladas por Thevenin [DT99] y por Lozano [LGR01].

THEVENIN La herramienta genérica presentada por Thevenin ([DT99],1999) es una aproximación para el desarrollo de interfaces de usuario plásticas. La plasticidad

es la capacidad de la interfaz para resistir variaciones, tanto en las características físicas del sistema como en el entorno, preservando su usabilidad. Esta herramienta puede mejorar los generadores de interfaces de usuario basados en modelos actuales que tienen en cuenta aspectos de usabilidad, ayudando a que satisfagan los requerimientos impuestos por la plasticidad. La herramienta consta de siete componentes:

- Modelo de Tareas de Usuario: representa las actividades del mundo real (modelos mental y de tareas).
- Modelo de Tareas del Sistema: describe cómo el sistema alcanza las tareas conceptuales mediante secuencias de tareas de interfaz (modelo de diálogo)
- Interfaz de Usuario Abstracto: representación del rendering abstracto de los conceptos y funciones del dominio (modelo de aplicación)
- Modelo de Interadores: describe los interadores disponibles para el rendering físico de una interfaz particular (Modelo de Presentación Abstracto)
- Modelo de Plataforma: describe las características físicas de las plataformas finales, representadas en términos de recursos físicos cuantificados: dispositivos (ratón, monitor, etc), capacidades (memoria, procesador, etc) y facilidades de comunicación (ancho de banda de los canales)
- Modelo de Entorno: especifica el contexto de uso englobando objetos, usuarios y eventos que puedan afectar al comportamiento
- Interfaz de Usuario Física: se obtiene como resultado de la resolución y/o propagación de las restricciones impuestas en los componentes Modelo de Plataforma, de Interfaz de Usuario Abstracto, de Interadores y de Entorno.

IDEAS IDEAS, de Lozano et al. ([LGR01], 2001), propone un entorno de desarrollo de interfaces de usuario basado en modelos dentro del proceso de desarrollo de software OO. Hace uso de técnicas de casos de uso y análisis de tareas y tiene en cuenta criterios de usabilidad. El modelo que propone refleja la interacción del usuario con el sistema y proporciona una representación formal del diseño de la interfaz. Incluye la generación de diagramas gráficos para representar los diferentes aspectos de la interfaz y permite la generación automática de la interfaz final a partir de los modelos definidos. El generador, aplicando técnicas de casos de uso y análisis de tareas,

construye los diferentes modelos que constituyen el modelo de interfaz de usuario, donde cada uno de ellos representa diferentes aspectos involucrados en la construcción de la interfaz. Así:

- A nivel de análisis, una vez definido el modelo de casos de uso se genera el Modelo de Tareas, identificando para cada caso de uso la(s) tarea(s) correspondiente(s).(Modelo de Tareas)
- A continuación se realiza el Modelo de Dominio, dado por los Diagramas de Secuencia asociados a cada caso de uso, y el Modelo de Roles que define las clases que intervienen en los diagramas anteriores y sus relaciones.
- Posteriormente, se define el Modelo de Usuario donde se indica, para cada usuario, las tareas y acciones permitidas en la interfaz. (Modelo de Aplicación y parte del Modelo de Diálogo)
- A nivel de diseño se elabora el Modelo de Diálogo formado por el Diagrama de Interacción de Diálogos, que indica las posibles transiciones de ventanas o diálogos, y el Diagrama de Especificación de Componentes, que establece, para cada ventana o diálogo, el conjunto de herramientas de control y visualización necesarias para poder llevar a cabo las tareas de usuario.(Modelo de Presentación Abstracto y parte del Modelo de Diálogo)
- En el ámbito de la implementación se genera el Modelo de Presentación que especifica los objetos de interacción concretos que formarán parte de la interfaz gráfica. En este momento el diseñador deberá tener presente diferentes guías de estilo y heurísticas que le permitan mejorar la usabilidad de sus diseños.(Modelo de Presentación Concreto)

A partir de los modelos de presentación se generan los prototipos de las interfaces, que son evaluados (test de usabilidad) por los usuarios finales de la aplicación.

Todo lo anterior se integra dentro de un proceso de diseño iterativo que permite incorporar las sucesivas mejoras que los usuarios introducen en el proceso de diseño.

La utilización de los modelos en las herramientas podemos verla en las tablas 3.6 3.7, 3.8 y 3.9, indicando el tipo de utilización final que se da al modelo. [Sil00] realiza la clasificación haciendo una simplificación de los modelos de tareas y diálogo en un

Herramienta	Utilización del Modelo
ADEPT	Modelo del Problema
AME	Modelo de Aplicación
FUSE	Modelo de Dominio del Problema
HUMANOID	Diseño de semánticas de la Aplicación
JANUS	Dominio del Problema
ITS	Pool de datos
MASTERMIND	Modelo de Aplicación
MECANO	Modelo de Dominio
TADEUS	Modelo de Dominio del Problema
TEALLACH	Modelo del Dominio
TRIDENT	Modelo de Aplicación
THEVENIN	Interfaz abstracto
IDEAS	Modelo de Usuario

Tabla 3.6: Utilización del Modelo de Aplicación en las herramientas MB-UIDE

mismo modelo, el modelo tarea-diálogo, al considerar que se trata del mismo concepto aunque cada uno de los modelos lo tratan a diferente nivel de abstracción.

Para poder determinar la incorporación de alguna funcionalidad correspondiente al modelo de composición a nivel abstracto y concreto vamos a examinar los constructores que incorporan las herramientas, centrándonos ya en concreto en los modelos de presentación, tanto abstractos como concretos. [Sil00] determina para los Modelos de Presentación Concreto y Abstracto los constructores que se reflejan en la tabla 3.11. En la tabla 3.10 pueden verse todos los constructores para los Modelos de Presentación Abstrato y Concreto. Los constructores que no están presentes en la tabla 3.11 no se usan en las herramientas o bien no están identificados en la literatura.

3.7 Conclusiones del estudio de las herramientas

Al observar la tabla 3.11, y dentro de las herramientas que inicialmente utilizan constructores para los Modelos de Presentación Abstracto y Concreto podemos concluir lo siguiente:

- Sólo existen cuatro herramientas que incorporan dentro del modelo concreto de

Herramienta	Utilización del Modelo
ADEPT	Modelo de Tareas
AME	OOD (asumo Object Oriented Domain)
FUSE	Modelo de Tareas
HUMANOID	Manipulación, secuenciación, efectos de acción
JANUS	Ninguno
ITS	Especificación de control sobre el diálogo
MASTERMIND	Modelo de Tareas
MECANO	Modelo de Tareas de Usuario / Modelo de Diálogo
TADEUS	Modelo de Tareas / Diálogo de Navegación
TEALLACH	Modelo de Tareas
TRIDENT	Modelo de Tareas
THEVENIN	Tareas Usuario / Tareas Sistema
IDEAS	Modelo de Tareas / Modelo de Diálogo

Tabla 3.7: Utilización del Modelo de Tarea-Diálogo en las herramientas MB-UIDE

Herramienta	Utilización del Modelo
ADEPT	Modelo de Interfaz de Usuario Abstracta
AME	OOA (asumo Object Oriented Aplication)
FUSE	Interfaz de Usuario lógica
HUMANOID	Presentación
JANUS	No Determinado (surveyed)
ITS	Especificación de marcos en el diálogo
MASTERMIND	Ninguno
MECANO	Ninguno
TADEUS	Procesamiento de Diálogo
TEALLACH	Modelo de Presentación
TRIDENT	No Determinado (surveyed)
THEVENIN	Interador
IDEAS	Diálogo / Presentación

Tabla 3.8: Utilización del Modelo de Presentación Abstracto en las herramientas MB-UIDE

Herramienta	Utilización del Modelo
ADEPT	Prototipo de interfaz
AME	Prototipo
FUSE	Interfaz de Usuario
HUMANOID	Presentación
JANUS	No Determinado (surveyed)
ITS	Especificación de estilo
MASTERMIND	Modelo de Presentación
MECANO	Modelo de Presentación
TADEUS	Procesamiento de diálogo
TEALLACH	Modelo de Presentación
TRIDENT	Modelo de Presentación
THEVENIN	No hay indicación
IDEAS	Modelo de Presentación

Tabla 3.9: Utilización del Modelo de Presentación Concreto en las herramientas MB-UIDE

Modelo	Constructor	Abreviatura	Función
APM	view	VIEW	Una colección de AIO's agrupados de forma lógica para guiar las entradas y salidas de una tarea
APM	Abstract Interaction Object	AIO	Un objeto de la interfaz de usuario sin representación gráfica e independiente de cualquier plataforma
CPM	window	WINDOW	La representación visible y manipulable de una view
CPM	Concret Interaction Object	CIO	una representación visible y manipulable de los objetos de interfaz de usuario que puede ser usado para manejar la información de entrada/salida del usuario asociada a una tarea
CPM	layout	LAY	Un algoritmo que provee de la colocación de CIO's en windows

Tabla 3.10: Constructores pertenecientes a los modelos abstracto y concreto de presentación

Herramienta	Constructor	Nombre
ADEPT	AIO	User Interface Object
ADEPT	CIO	UIO
AME	AIO	AIO
AME	WINDOW	OOD Class
AME	CIO	CIO
AME	LAY	layout-method
HUMANOID	AIO	template
HUMANOID	WINDOW	display
HUMANOID	CIO	display, interaction technique
HUMANOID	LAY	layout
JANUS	CIO	interation object
JANUS	WINDOW	dialog window (UIView)
ITS	WINDOW	root unit
ITS	CIO	unit
ITS	LAY	style attribute
MASTERMIND	WINDOW	presentation
MASTERMIND	CIO	presentation part
MASTERMIND	LAY	guides, grids, conditional
TEALLACH	VIEW	free container
TEALLACH	AIO	AIO
TEALLACH	WINDOW	window
TEALLACH	CIO	CIO
THEVENIN	AIO	Interador

Tabla 3.11: Utilización de Constructores pertenecientes a los modelos de presentación

presentación un sistema de representación de la composición de componentes de la interfaz de usuario (AME, HUMANOID, ITS y MASTERMIND).

- Sólo dos de las herramientas (ITS y TEALLACH) contemplan la posibilidad de reflejar, en el Modelo de Presentación Abstracto, el agrupamiento de componentes como representación semántica y de organización lógica de los componentes. Sin embargo, todas las herramientas, a excepción de ADEPT, contemplan la representación de dicha agrupación a nivel de Modelo de Presentación Concreto; esto es, mediante un componente de agrupación de componentes visible y manipulable.

- Ninguna de ellas contempla la representación abstracta de la composición, ya que aquellas que poseen un constructor de colocación de componentes utilizan, en la mayoría de los casos, sistemas de estilos o algoritmos de colocación, excepto quizás HUMANOID.

- No existe en ninguna la representación abstracta de los componentes del Modelo de Presentación Concreto que permita independizar la presentación visual final de la plataforma para la cual se definen.

3.8 Conclusiones del capítulo

Dada la importancia de la componente visual de la interfaz se han estudiado los modelos y herramientas más importantes para la especificación o representación de la interfaz de usuario.

Una vez examinados podemos concluir que existen muy poca literatura acerca de la representación de la componente visual de la interfaz. En algunos modelos se contempla la presentación abstracta de la interfaz pero no la concreta, con lo cual esta parte está supeditada a la opinión de un implementador, y por lo tanto puede no reflejar todas las características de la interfaz identificadas por el modelo.

La mayoría de los modelos que incorporan representación abstracta de la interfaz lo hacen a nivel de comportamiento, y solo unos pocos realizan una abstracción de la representación concreta de la interfaz (a nivel de visualización). Prácticamente todos los que incorporan representación concreta de la interfaz es porque incorporan herramientas de generación automática de código de interfaz.

Una idea clara que se obtiene después del estudio de los modelos y herramientas es que su foco de diseño es principalmente la funcionalidad de la interfaz. El papel del usuario en el diseño es para indicar su forma de pensar sobre el proceso a desarrollar y determinar las tareas que le son interesantes. La propuesta que se desarrolla en este trabajo se centra en la definición de la interfaz desde su uso por parte del usuario, y es éste el que, con sus indicaciones de necesidades sobre la interfaz visual, determina qué componentes deben existir y qué comportamiento deben poseer. Ambas visiones de la interfaz confluyen en un punto medio y, sin duda, pueden unirse. Esta unión se plantea en el capítulo 6 como un trabajo futuro a realizar.

Es muy importante resaltar que los modelos que realizan una representación abstracta o concreta lo hacen a nivel de componente, y solo uno de ellos (PUM) hace referencia a una representación de la colocación de los componentes dentro de la misma. Esta falta de representación de la composición de los componentes para generar la estructura final de la interfaz, junto con la necesidad de contemplar el comportamiento dinámico actual de los componentes, muestran una deficiencia en los modelos y herramientas que este trabajo pretende solucionar.

Ninguna herramienta proporciona mecanismos para su representación a partir de la componente visual que el usuario percibe. No permite la exportación de la definición

realizada de la interfaz y se realiza para una plataforma concreta.

Capítulo 4

Representación abstracta de la interfaz de usuario

La interfaz de usuario es un estructura eminentemente visual por lo que su diseño no debería estar sujeto a las restricciones impuestas por las herramientas de prototipado y entornos de programación existentes.

A continuación se propone un sistema mediante el cual un diseñador gráfico, o incluso un usuario, pueda realizar una definición de su interfaz de usuario utilizando para ello aspectos estéticos.

Esta definición de la interfaz de usuario es absolutamente libre, y permite dotar a los diseñadores de aplicación de un sistema de representación independiente de la plataforma de desarrollo y que incluye tanto la componente visual de la interfaz de usuario como su comportamiento.

Consideraremos que la interfaz de usuario visual no es una estructura continua sino que está compuesta por elementos finitos discretos. Esto es concordante con toda la literatura sobre interfaces de usuario. Plantearemos la interfaz de usuario como una composición de elementos individuales a los cuales denominaremos componentes. Los componentes siguen una estructura topológica jerárquica, de forma que pueden estar contenidos unos dentro de otros, teniendo siempre en cuenta que, como se ha comentado en capítulos anteriores, la interfaz de usuario visual se visualiza en un dispositivo de salida con un tamaño y resolución finitos. Para la definición de la

interfaz de usuario se dota al diseñador gráfico de un sistema de representación en el cual puede:

- Realizar la definición visual de los componentes de la interfaz de usuario, posibilitando la creación de jerarquías de composición (un componente dentro de otro), y la definición de nuevos tipos de componentes a nivel de metáfora o composición.
- Indicar la composición de componentes individuales para formar la interfaz de usuario concreta con la que el usuario interaccionará.
- Representar el diálogo entre componentes dentro de la interfaz de usuario, indicando los eventos a los que cada componente atiende y de qué manera la interfaz de usuario responde a los mismos, ya sea modificando los componentes de la interfaz o bien realizando llamadas a la aplicación.

Se realiza una distinción de dos casos en la propuesta, debido a que cada uno de ellos tiene un funcionamiento distinto y eso se refleja en la representación. Primero se define una interfaz estática a nivel de parámetros visuales referentes a la capacidad por parte del usuario de realizar una serie de operaciones sobre el aspecto visual de los componentes, como modificación de su tamaño y posición sobre el dispositivo de visualización, y sobre la jerarquía visual, entendida ésta como la cercanía al usuario de un componente u otro.

Una interfaz dinámica, por el contrario, permite la realización de operaciones visuales sobre los componentes de la interfaz de usuario. La actuación de estas operaciones sobre la apariencia visual de los componentes modifica las capacidades de la interfaz a la hora de establecer el diálogo con el usuario y de permitir una correcta comunicación visual con el mismo.

Entenderemos por **"operación visual"** como aquella que permite modificar aspectos visuales de los componentes como su tamaño, su colocación en el dispositivo de visualización o el orden de presentación visual de los componentes. Permite colocar un componente de la interfaz en una situación más cercana al usuario, y por lo tanto en su centro de atención, sin que así esté establecido en el diálogo de componentes de la interfaz.

En el caso estático de la interfaz de usuario, los componentes tienen una posición

fija en el dispositivo de visualización y con un tamaño predeterminado. El único diálogo permisible al usuario es el indicado en la definición de los componentes de la interfaz, con lo cual no existirán estados de la interfaz que dependan de una actuación arbitraria del usuario.

Entenderemos como **"estado de la interfaz"** la conjunción de los componentes que forman la interfaz con la que el usuario interactúa en un momento concreto.

Dada la naturaleza no arbitraria de las actuaciones por parte del usuario, el conjunto de estados de la interfaz es finito y puede ser descrito y evaluado. Un estado depende de los componentes que posee junto con las propiedades de cada uno de ellos. Un estado de la interfaz corresponde con una situación de la interfaz en la cual está esperando por una acción del usuario, y no varía mientras el usuario no interactúe con la misma.

4.1 Componentes de la interfaz de usuario

Un componente de la interfaz de usuario es un elemento individual de la misma que es perceptible por el usuario a nivel de apariencia visual, responde a una acción del usuario, o bien sirve de elemento agrupador de otros componentes.

Cada componente de la interfaz de usuario tiene unas características propias relacionadas con su apariencia visual y que son independientes de la interfaz. Estas características se consideran inherentes al componente. Es habitual que existan componentes de aspecto igual en interfaces de usuario de funcionalidad diferente. El usuario está acostumbrado a que el componente para la realización de alguna acción tenga una apariencia en concreto, aunque la aplicación asociada con la interfaz tenga un propósito distinto. Por ejemplo, todos los componentes correspondientes a la opción de "submit"en las páginas html, en su sección de formularios, son iguales. Otro caso muy representativo son los widgets de los toolkits de las herramientas de construcción de interfaces de usuario. Varios componentes con un comportamiento particular se agrupan siempre con la misma apariencia visual y un comportamiento dentro de esa apariencia visual semejante.

Además, los componentes poseen unas propiedades relativas a su posicionamiento y comportamiento que ya no son inherentes al componente sino que se derivan de su

participación o papel dentro de la interfaz de usuario en la cual están incluidos.

Por ello consideraremos la información del Componente Visual Abstracto (CVA) compuesta por :

- Su geometría. Se construye en base a tres posibles descripciones dependiendo del tipo de componente que se trate: componentes gráficos (primitivas gráficas junto con operaciones sobre estas primitivas para construir una representación compleja), componentes de texto (primitivas de texto para su representación), y componentes gráficos complejos (primitivas por enumeración (imágenes almacenadas)).
- El conjunto de atributos que determinan la apariencia visual de las primitivas de la geometría.
- El conjunto de propiedades que el Componente Visual Abstracto posee con respecto a su posición y tamaño.

Definición Denominamos Componente Visual Abstracto (CVA) a la representación abstracta de un objeto de la interfaz, el cual puede ser representado en un dispositivo de visualización mediante la definición de dicha representación abstracta.

Definición Denominamos Componente Visual Concreto (CVC) a la representación en un dispositivo de visualización de un Componente Visual Abstracto.

Premisa Todo Componente Visual Abstracto posee asociado un Componente Visual Concreto, sea o no perceptible por el usuario.

4.1.1 Geometría y Apariencia

Como se ha indicado en la sección anterior, se define la geometría de los componentes, y sobre dicha geometría se definen los atributos que proporcionan la apariencia visual perceptible por el usuario. Dado que los componentes no serían perceptibles sin sus atributos de apariencia, y que no tiene sentido la utilización de dichos atributos sin una geometría sobre los que aplicarlos, a partir de ahora cuando se haga referencia a

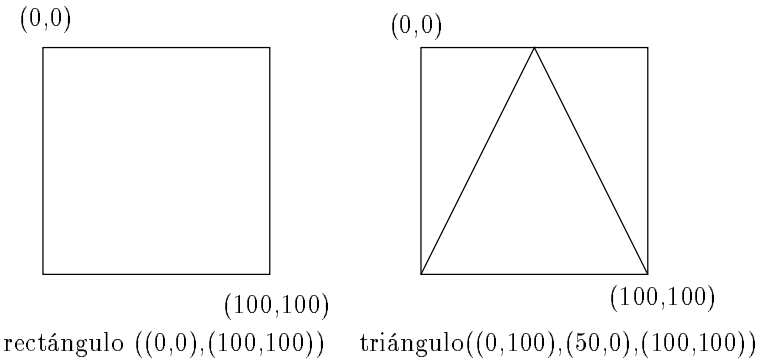

Figura 4.1: Punto de referencia en figuras regulares e irregulares

la apariencia de un componente se hará a la conjunción de geometría y atributos que determinan su apariencia.

Como primer paso para la definición de los componentes es necesario expresar, un sistema de coordenadas que nos permita utilizar la geometría para la definición de los componentes y su composición.

El sistema de coordenadas para la identificación y dibujado de elementos en la pantalla es una de las decisiones más importantes a tomar dentro de un sistema de visualización. El más utilizado es aquel en el cual el origen de coordenadas está en la esquina superior izquierda.

La geometría del objeto se construye mediante una función de referencia a un punto del objeto que se denominará punto de referencia, que corresponderá con una coordenada (x, y), donde para el objeto situado en un plano con un eje de coordenadas positivo ($x >= 0$ e $y >= 0$), $x = 0$ corresponderá con el menor punto en el eje X de las primitivas, e $y = 0$ corresponderá con el mayor punto del eje Y de las primitivas.

En figuras regulares, como por ejemplo un rectángulo, el punto de referencia coincidirá con el punto p_0, como en el caso de una línea. Sin embargo, en figuras no regulares la determinación del punto de referencia debe hacerse estudiando el valor de sus puntos de definición. Por ejemplo, en la figura 4.1 se observa un triángulo dado por sus vértices polígono(p_0, p_1, p_2); el punto de referencia (x, y) corresponde con $x = p_i(x)$, donde p_i es el menor de $p_0(x), p_1(x)$ y $p_2(x)$, e $y = p_i(y)$, donde p_i es el menor de $p_0(y), p_1(y)$ y $p_2(y)$.

Premisa Un Componente Visual Abstracto tiene como punto de referencia el dado por el dispositivo de visualización en el cual será representado.

Dentro del conjunto de primitivas, y con el objeto de poder garantizar una relación biunívoca entre el Componente Visual Abstracto y su representación en el dispositivo de visualización, debe incluirse también la primitiva de imagen que en el momento de la proyección dará lugar a una proyección por enumeración. Esto es debido a que para imágenes complejas no puede garantizarse su unicidad en el dispositivo de visualización, al depender de la técnica de representación utilizada. Por lo tanto, para aquellos CVA que requieran de la construcción de primitivas de mayor nivel que las básicas, o si el conjunto de las mismas necesario para definirlo es excesivo, lo haremos a través de una primitiva de imagen.

Para establecer el conjunto de primitivas gráficas a utilizar para determinar la geometría de cada CVA utilizaremos un conjunto genérico fácilmente transformable a cualquier paquete gráfico de implementación, como el de la tabla 4.1. El conjunto de primitivas aquí descrito para la representación de la geometría de los componentes no es un conjunto cerrado para la definición de componentes de la interfaz. Se ha elegido este conjunto de primitivas simples en dos dimensiones debido a dos razones. Por un lado, la mayoría de las interfaces de usuario actuales son en dos dimensiones y media en base a los actuales estilos de interacción en interfaces de manipulación directa. Las dos primeras dimensiones son el ancho y alto de componentes y la media dimensión restante es la dada por el solapamiento de componentes en el dispositivo de visualización. Este solapamiento es identificado en la representación como la jerarquía visual de los componentes de la interfaz. Por otro lado, es un conjunto de primitivas que se identifica como el conjunto mínimo necesario para la construcción de cualquier otra primitiva más compleja. Está contemplado en todos los sistemas de gráficos para la generación de visualización de elementos gráficos, con lo cual se garantiza la exportabilidad de la representación. Lo mismo se debe indicar con respecto a los atributos que determinan la apariencia de la geometría de los componentes. No es un conjunto cerrado, pero es el suficiente para una representación de la interfaz que permita un uso claro y sencillo de la misma, y el desarrollo de herramientas de generación automática de código de interfaz en diferentes entornos de desarrollo.

Puede extenderse la representación del presente trabajo con cualquier conjunto de primitivas y atributos para la representación de componentes de la interfaz de usuario que respondan a otro tipo de geometría, como por ejemplo componentes en tres

Primitiva	Descripción
Línea(p_0, p_1)	p_0 punto de comienzo de línea y p_1 punto final de línea
Rectángulo(p_0, p_1)	p_0 corresponde con la esquina superior izquierda y p_1 con la esquina inferior derecha
Polígono($p_0, p_1, ..., p_n$)	Los puntos se corresponden con los vértices del polígono
Elipse(rectángulo (p_0, p_1), angulocomienzo, angulofinal)	El rectángulo corresponde con el menor en que la elipse está inscrita. p_0 y p_1 determinan la altura y anchura de la extensión de la figura de la elipse. Si el arco de la elipse está cerrado o no viene determinado por los ángulos de comienzo (angulocomienzo) y ángulo de finalizacion (angulofinal)
Imagen(*imagen*)	Primitiva de imagen que tendrá su geometría asociada a un archivo
Texto(*texto*)	Donde la colocación del texto dentro de la interfaz vendrá dada por el rectángulo que englobe dicho texto.

Tabla 4.1: Primitivas gráficas y de texto

dimensiones. Entre otras consideraciones, una interfaz de usuario de este tipo debe incorporar un proceso de proyección previo a su visualización (en el caso de un dispositivo de visualización clásico) o bien tener en cuenta en la representación un nuevo dispositivo de visualización e interacción. Una interfaz de usuario de otra naturaleza tiene que tener en cuenta las diferencias a la hora de incorporar nuevos eventos para el diálogo de la interfaz. Ya que, con toda seguridad, los análisis planteables sobre la misma serán diferentes de los aquí mostrados.

En la tabla 4.2 se describen una serie de atributos a nivel general de abstracción que tendrán aplicación sobre las primitivas descritas anteriormente para alterar su apariencia sobre una geometría de componente.

Las primitivas de texto se han incluido en la tabla con el resto de primitivas gráficas

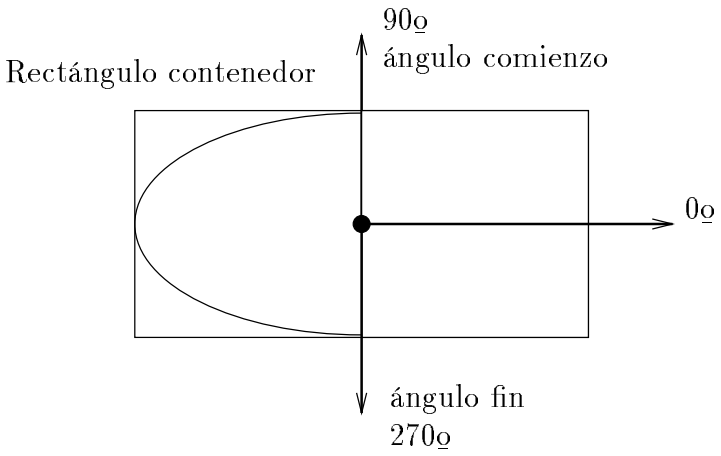

Figura 4.2: Especificación de arcos elípticos

por encontrarse en la totalidad de los paquetes gráficos. En la tabla 4.3 pueden observarse algunos de los atributos aplicables a las primitivas de texto.

La apariencia de cualquier componente puede reproducirse a partir de la representación aquí indicada, teniendo en cuenta que la complejidad de los valores de los atributos que se especifiquen deberá ser satisfecha por los paquetes gráficos que implementen la especificación sobre un determinado dispositivo de visualización.

Se ha decidido establecer una representación de componentes diferente en función de su tipo (componente gráfico, de texto o por enumeración) pues optar por utilizar un único formato provocaría la existencia de muchos elementos de información sin valor. En las tablas 4.4, 4.5 y 4.6 se muestran los formatos de representación de la apariencia para cada uno de los distintos componentes.

Atributo	Descripción
EstiloLínea()	Determina la forma de la línea. Por defecto será "continuo"
AnchoLínea()	Habitualmente el ancho de línea se mide en píxeles de pantalla. Por defecto será 1
ColorLínea()	Asigna un color a la primitiva de geometría
ColorRelleno()	Se indica la primitiva a rellenar y el atributo de relleno. Dentro de estos atributos podríamos encontrar "sólido", "opaco", "transparente", ...

Tabla 4.2: Atributos aplicables a las primitivas gráficas

Primitiva	Descripción
Fuente()	Fuente a utilizar
TamañoFuente()	Tamaño a utilizar de la fuente seleccionada
Colorfuente()	Color a aplicar a la fuente

Tabla 4.3: Atributos aplicables a las primitivas de texto

Item	Tipo	Descripción
COMP:	Ob	Cadena identificación comienzo descripción componente
G	Ob	Carácter de identificación de componente gráfico
#	Ob	Separador
<primitivas>	Ob	Primitivas que definen el componente
<primitivas>	Ob	Opciones de primitivas
PRIM:	Ob	Cadena identificación comienzo descripción primitiva
<primitiva>	Ob	Puede coger los valores : Rectángulo(p0, p1), línea(p0, p1), círculo(p0, radio), elipse(rectángulo(p0, p1), AngCom, AngFin), polígono(p0, p1, . . . ,pn)
#	Ob	Separador
EstiloLínea=valor	Ob	Estilo de línea de la primitiva. Por defecto continuo
#	Ob	Separador
AnchoLínea=valor	Ob	Tamaño en valor píxeles de la línea de la primitiva
#	Ob	Separador
ColorLínea=valor	Ob	Color de la línea de la primitiva especificado en RGB XXXXXX
#	Ob	Separador
ColorRelleno=valor	Ob	Color de relleno de las primitivas especificado en RGB XXXXXX

Tabla 4.4: Formato de descripción de la apariencia de componentes gráficos

Item	Tipo	Descripción
COMP:	Ob	Cadena identificación comienzo descripción componente
T	Ob	Carácter de identificación de componente textual
#	Ob	Separador
<primitivas>	Ob	Primitivas que definen el componente de texto
#	Ob	Separador
Fuente(valor)	Ob	Indicación de la fuente que utiliza la primitiva de texto
#	Ob	Separador
TamañoFuente=valor	Ob	Tamaño de la fuente de texto utilizada
#	Ob	Separador
ColorFuente=Valor	Ob	Color aplicado al texto

Tabla 4.5: Formato de descripción de los componentes de texto

Item	Tipo	Descripción
COMP:	Ob	Cadena identificación comienzo descripción componente
B	Ob	Carácter de identificación de componente por enumeración
#	Ob	Separador
<fichero>	Ob	Nombre del archivo donde se encuentra la descripción del componente por enumeración

Tabla 4.6: Formato de descripción de los componentes gráficos por enumeración

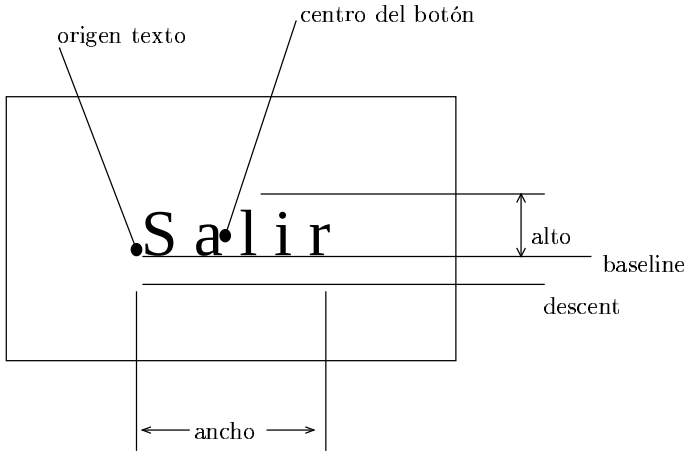

Figura 4.3: Apariencia del componente

Un ejemplo de representación de dos componentes sería, por ejemplo, un componente gráfico con un componente texto interno, que en la mayoría de las interfaces de usuario tendría una funcionalidad de ejecución de tarea.

Componente gráfico	Componente texto
Rectángulo(40,20)	Texto("Botón")
EstiloLínea=Continuo	FuenteTexto=SansSerif
AnchoLínea=1	TamañoFuente=12
ColorLínea=000000	ColorFuente=000000
ColorRelleno=FFFFFF	

De este modo se representa la apariencia visual de los dos componentes, aunque no la colocación de un componente dentro de otro, tal y como se muestra en la figura 4.3. Las propiedades de posición y tamaño se desglosan en la sección siguiente.

4.2 Propiedades de posición y tamaño

En la sección anterior hemos realizado una representación abstracta de la parte visual de un componente de la interfaz de usuario. En este momento podemos reproducir su

aspecto, pero no se ha definido todavía cuál es su colocación dentro del dispositivo de visualización, o si está dentro de otro componente, ni si incluye a otros componentes de la interfaz.

Para poder representar esta colocación o topología vamos a examinar los distintos casos que se pueden presentar.

Cualquier interfaz de usuario se caracteriza por poseer un único punto de inicio, partida o estado inicial, en el cual se muestra la información necesaria para que el usuario pueda manejarla, y se espera a que el usuario realice su primera acción.

Dado que el ser humano tiende a agrupar y realizar un ajuste óptico a nivel perceptual más que físico, como se ha visto en el capítulo 2, la mayoría de las interfaces de usuario de una aplicación tienden a estar agrupadas visualmente de forma jerárquica. Esto significa que existe un elemento visual que agrupa todos los componentes asociados a una interfaz. Dicho componente suele pertenecer al estado inicial de la interfaz de usuario y el resto de interacción con la interfaz tiende a circunscribirse a dicho componente inicial. Pueden existir componentes de interfaz de la aplicación que no sigan una inclusión topológica en el componente principal de la interfaz, pero en la mayoría de los casos tendrán algún tipo de dependencia, ya sea visual (a nivel de atributos similares) o de interacción (un componente necesario para seguir interactuando con la interfaz de usuario de la aplicación)

Por todo ello, un Componente Visual Abstracto puede tener restricciones de relación con el dispositivo de visualización en el cual va a ser representado, o con el componente en el cual está incluido. Vamos a distinguir dos tipos de restricciones: de tamaño y de posición.

4.2.1 Propiedad de posición

En un primer momento se podría pensar que las propiedades topológicas o de colocación relativa entre componentes son más importantes que las propiedades de posición, pero realmente unas son inherentes a las otras. De esta forma, una propiedad topológica, por ejemplo un centrado, tiene una relación única para un tamaño de componente contenedor y un componente contenido. El caso es extremo para una interfaz de usuario estática en la cual el usuario no puede modificar la propiedad de tamaño de ninguno de los componentes, pues la posición del componente contenido será siempre

la misma.

Las propiedades de posición se refieren a la posición de ese CVA dentro del tamaño que tiene asignado el componente en el cual está contenido, o bien en el dispositivo de visualización si no está contenido dentro de ningún otro componente, y son:

- Fija. Se da una posición concreta sobre el sistema de coordenadas del dispositivo de visualización o bien sobre el componente que lo contiene.
- Relativa. Se indica una propiedad topológica a mantener siempre entre el componente contenido y el componente contenedor. Con el objeto de no tener que calcular en todos los casos la propiedad de posición del componente contenido dentro del contenedor, y dado que en el estado inicial de la interfaz la posición está predefinida, se incluye también en esta propiedad la posición concreta mediante coordenadas. Las posibles propiedades de posición relativa son: Centrado(V, H, A) (Vertical, Horizontal o Ambos) y Justificado(De, Iz, Su, In) (Derecha, Izquierda, Superior, Inferior) con su correspondiente valor de justificación en píxeles.

Las propiedades de posición relativa se incluyen en el propio componente por ser inherentes al componente individual. Más adelante se mostrarán otras propiedades de posición que no pueden estar en la definición del componente individual por lo que se colocarán en el componente contenedor.

En base a estas propiedades de posición se amplían los elementos de la representación con la introducción de estos nuevos conceptos.

Definición Denominamos Componente Contenedor a un Componente Visual Abstracto que contiene otros Componentes Visuales Abstractos.

Premisa La interfaz de usuario visual tiene siempre como mínimo un componente contenedor que contiene a todos los demás componentes, o en su defecto se considerará como componente contenedor el dispositivo de visualización.

Premisa Los componentes contenedores no tienen capacidad de interacción con el usuario, se emplean únicamente como contenedores de otros componentes.

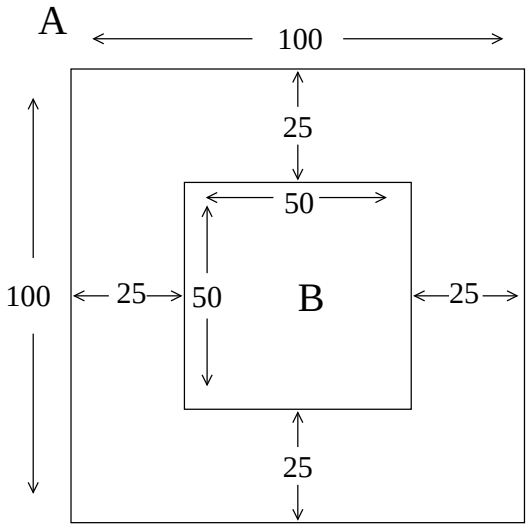

Figura 4.4: Componente B con Centrado(A) (Ambos: Vertical y Horizontal)

Premisa Si un componente contenedor no está dentro de ningún otro componente contenedor se considerará su punto de referencia como el punto de referencia del sistema de visualización que se esté utilizando.

En caso de interfaz estáica, como la que estamos tratando, coinciden los valores de posición fija y relativa de los componentes en cualquier momento de la utilización de la interfaz (Figura 4.4). Esto es debido a que, al no producirse modificaciones en el tamaño de los componentes contenedores, no se modifican las posiciones de los elementos contenido. Por ello, en el caso estático no es necesario utilizar las posiciones lógicas. Esto mismo puede aplicarse a la propiedad de justificado.

El formato de representación para la propiedad de posición de un componente individual es el mostrado en la tabla 4.7.

Existen otras propiedades de posición aplicables a un componente pero que no tienen significación sin otros componentes del mismo nivel. Estas propiedades son el Equiespaciado y el Alineado. La diferencia fundamental que existe con las propiedades de posición vistas hasta el momento radica en que la aplicación de las nuevas necesita tener la indicación de todos los componentes que intervienen en la propiedad. Por ello, esta descripción de la propiedad posición se realiza en el componente contenedor, indicando los componentes contenido que están implicados y el valor común que todos ellos utilizan.

Item	Tipo	Descripción
Posicion(OpcionesP)	Ob	Posición del componente dentro de su componente contenedor o bien dentro del dispositivo de visualización
OpcionesP Fija(x,y) Relativa (OpcRel)(x,y)		Opciones correspondientes a la posición del componente. Estas pueden ser: Posición fija del componente dentro del componente contenedor, siendo (x,y) su posición superior izquierda Posición relativa indicada mediante restricciones topológicas. (x,y) es el valor inicial de posición sobre su componente contenedor
OpcRel Centrado(H,V,A) Justificado (De(valor), Iz(valor), Su(valor), In(valor)) Superior		Opciones de posición relativa Centrado del componente en su componente contenedor en Horizontal, Vertical o Ambas Justificado del componente en su componente contenedor a la Derecha, Izquierda, Superior o Inferior Su posición viene determinada por la topología indicada en el componente contenedor

Tabla 4.7: Formato de descripción de la propiedad posición para componentes contenido a nivel individual

Item	Tipo	Descripción
COMP:	Ob	Cadena identificación comienzo composición componente
<nombre de componente>	Ob	Nombre del componente contenedor que se define
#	Ob	Separador de componente contenedor e información de su contenido
Alineado(Opciones)	Op	Los componentes contenidos están alineados
Opciones Iz(valor) De(valor) Su(valor) In(valor)		Opciones en Alineado Alineado a la izquierda a una distancia de valor en píxeles Alineado a la derecha a una distancia de valor en píxeles Alineado arriba a una distancia de valor en píxeles Alineado abajo a una distancia de valor en píxeles
Equiespaciado	Op	Los componentes contenido están equiespaciados entre sí y respecto a los bordes del componente contenedor para cada eje.
(<nombres de componentes>)	Ob	Nombres de los componentes contenido que forman parte de un posicionamiento, separados por #

Tabla 4.8: Formato descripción propiedad posición del conjunto de componentes de interfaz

La descripción de las propiedades de posición Equiespaciado y Alineado se muestran en la tabla 4.8. Estas propiedades están asociadas a la inclusión dentro de un componente contenedor de componentes contenido que tienen iguales características de posición. Por ello, las propiedades de posición de Equiespaciado y Alineado se colocan en la descripción de la composición de los componentes de la interfaz. En dicha descripción se indica cuáles son los componentes contenido para cada componente contenedor, y en que casos los componentes contenido guardan una relación común en la composición.

Las propiedades de Equiespaciado y Alineado conllevan la aparición de componentes contenedores que no poseen ninguna funcionalidad dentro de la interfaz, ni a nivel visual (puesto que no necesitan poseer ningún atributo perceptible por el usuario)

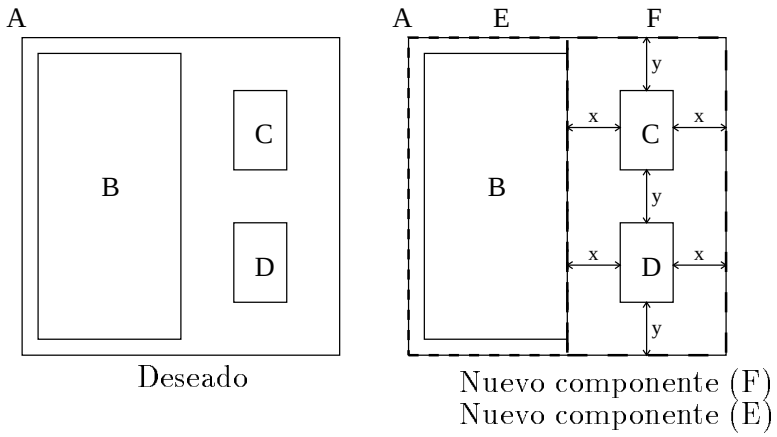

Figura 4.5: Introducción de componentes invisibles

ni a nivel de comportamiento. La única función que tienen es la de permitir una colocación común de otros componentes contenido. Puede verse un ejemplo en la figura 4.5. La interfaz deseada aparece a la izquierda. En ella están definidos cuatro componentes. El componente contenedor es A, y existen tres componentes contenido que son B, C y D. Puede realizarse una colocación manual indicando las posiciones numéricas que ocupan, o puede intentarse una colocación desde el componente contenedor. Esto puede resultar complejo ya que el objeto B debe situarse en la mitad izquierda del componente A, los componentes C y D en la mitad derecha y además están equiespaciados entre sí, con los bordes superior e inferior y con el borde derecho de A y el borde izquierdo del componente B. En este caso no se pueden indicar las propiedades de posición relativas con respecto al componente contenedor de A porque son relativas a partes del componente A y al componente B.

La solución a adoptar es la creación de dos componentes E y F que ocupen, respectivamente, la mitad izquierda y derecha del componente A. A su vez, el componente B puede justificarse a la derecha del componente E con valor 0, y los componentes C y D pueden equiespaciarse en el componente F.

Las propiedades de posición en los distintos componentes se representarían de la siguiente forma:

Descripción propiedad posición
COMP:A#Posición(Fija(x_1, y_1)
COMP:E#Posición(Justificado(Iz)(0)(x_2, y_2))
COMP:F#Posición(Justificado(De)(0)(x_3, y_3))
COMP:B#Posición(Centrado(V)Justificado(De)(0)(x_4, y_4))
COMP:C#Posición(Relativa(x_5, y_5))
COMP:D#Posición(Relativa(x_6, y_6))

Y las entradas correspondientes en el fichero de contenido:

Descripción propiedad composición
COMP:A#E#F
COMP:E#B
COMP:F#(Equiespaciado)(C#D)

4.2.2 Propiedad de tamaño

La propiedad correspondiente al tamaño de un componente, al igual que en el caso de la propiedad de posición, no es relevante en interfaces estáticas, en las cuales no existen operaciones visuales, y por lo tanto no existen modificaciones de tamaño en los componentes.

Aún así, atendiendo a la necesidad de contemplar interfaces dinámicas se incorpora en la representación la posibilidad de indicar tamaños relativos de un componente con respecto al componente superior.

La propiedad de tamaño del componente corresponde con el tamaño del menor rectángulo que puede contener las primitivas gráficas (en el caso de un componente gráfico), la imagen (en el caso de un componente de primitivas por enumeración) y las primitivas de texto (en el caso de un componente de primitiva de texto).

Las propiedades de tamaño se colocan en cada CVA, afectando al mismo únicamente. Estas propiedades de tamaño son de dos tipos:

- Fija. El tamaño del CVA está determinado de forma concreta y no puede modificarse.
- Relativa. El tamaño del CVA es dinámico de forma que es una proporción del tamaño del componente contenedor que lo contiene.

Item	Tipo	Descripción
Tamaño(OpcionesT)	Ob	Tamaño del componente dentro de su componente contenedor o bien dentro del dispositivo de visualización
OpcionesT		Opciones del tamaño relativo
Fijo(Tam_x,Tam_y)		Tamaño del componente de texto fijo en sus ejes x e y
Relativo(Tam_x,Tam_y)		Tamaño del componente relativo al tamaño del componente contenedor y con valores (Tam_x,Tam_y) iniciales

Tabla 4.9: Formato de descripción de la propiedad tamaño para componentes contenido

El formato de descripción de la propiedad de tamaño es el correspondiente a la tabla 4.9.

Veamos el ejemplo de la figura 4.4, en el que tenemos dos componentes A y B. La interfaz es estática, con lo cual no acepta modificaciones de tamaño, pero aún así podría colocarse como Relativa, ya que se indica explícitamente el tamaño inicial del componente dado un tamaño inicial del componente contenedor.

Descripción propiedad tamaño
COMP:A#Tamaño(Fija(100,100))
COMP:B#Tamaño(Fija(50,50))

lo cual es equivalente a:

Descripción propiedad tamaño
COMP:A#Tamaño(Relativa(100,100))
COMP:B#Tamaño(Relativa(50,50))

4.3 Soporte de las operaciones visuales en la representación

Ya se ha indicado la existencia de operaciones visuales que corresponden con acciones de usuario que modifican la apariencia de los componentes de la interfaz. Estas operaciones pueden corresponder con operaciones de cambio de posición o redimensionado de componentes en el dispositivo de visualización. La posibilidad de realización de estas operaciones visuales viene limitada por las propiedades de posición y tamaño de los componentes.

En el caso de que un componente contenedor independiente posea la propiedad de posición relativa, el usuario podrá realizar operaciones de cambio de posición sobre el mismo. Esta propiedad es útil en el caso de que durante el proceso de diseño de la interfaz se produzca un solapamiento de componentes activos en la jerarquía visual, porque posibilita el acceso del usuario a los componentes que están en el nivel jerárquico inferior por cambio de posición del componente de nivel jerárquico superior.

En el caso de que un componente contenedor independiente posea la propiedad de tamaño relativo, el usuario podrá realizar operaciones de modificación de tamaño de ese componente, y por defecto de todos los componentes contenido que posea y que tengan su tamaño relativo con respecto al componente contenedor.

A continuación se comprobará como se pueden reflejar las operaciones visuales sobre los componentes de la interfaz con el sistema de representación propuesto.

4.3.1 Validez de la propiedad de posición

La representación de la propiedad de posición en los componentes puede realizarse de dos formas diferentes: aplicada sobre un único componente en relación a su componente contenedor, o aplicada sobre múltiples componentes en relación a su componente contenedor.

En el caso de la relación en la propiedad de posición de un componente contenido con su componente contenedor, cabe recordar que cada componente tiene una posición inicial asignada sobre la cual podrían ejecutarse las operaciones visuales. Las operaciones visuales contempladas en el presente trabajo para los componentes contenedores son la modificación de posición del componente sobre el dispositivo de visualización y la

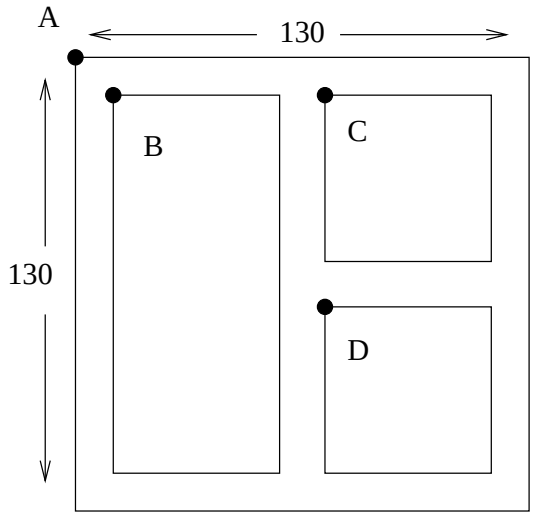

Figura 4.6: Componente ejemplo modificación posición y tamaño

modificación de tamaño o redimensionado del componente.

Sólo podrán realizarse operaciones visuales sobre componentes contenedores que posean su propiedades de posición y tamaño relativas, y dentro de estos sólo podrán realizarse sobre aquellos cuyas propiedades relativas no lo sean a otros objetos contenedores. Esto reduce la utilización de operaciones visuales a su uso sobre componentes contenedores independientes cuyo componente contenedor sea el dispositivo de visualización.

La propiedad de posición es muy dependiente de la propiedad de tamaño, puesto que una modificación de la propiedad de tamaño implica, en el caso de una propiedad de posición relativa, un recálculo de la posición del componente.

Para demostrar la validez de la representación sobre una interfaz de usuario dinámica que permita la operación visual de cambio de posición vamos a utilizar un ejemplo. En la figura 4.6 se observa un componente contenedor independiente con tres componentes contenido, cuyas posiciones son referentes al componente contenedor. Vamos a comprobar si al modificar la posición de A dentro del dispositivo de visualización se pueden obtener también las posiciones de B, C y D.

El componente contenedor tiene la posición (10, 10) dentro del dispositivo de visualización, y vamos a permitir su modificación hacia la posición (100, 100). La propiedad de Posición(Relativa)(10,10) del componente B nos indica que la posición de B con respecto a su componente contenedor A es (10,10). Si se desea obtener su posición en el dispositivo de visualización se realiza la siguiente operación:

$$Pos_{B_{cd}}(x) = Pos_{B_{cc}}(x) + Pos_{A_{cs}}(x) = 20$$
$$Pos_{B_{cd}}(y) = Pos_{B_{cc}}(y) + Pos_{A_{cs}}(y) = 20$$

donde:
cd=coordenadas de posición en el dispositivo del componente B,
cc=coordenadas de posición del componente B y
cs=coordenadas de posición del componente contenedor superior A

Se ha calculado la posición de B en el dispositivo a partir de su posición relativa con respecto a A.

En la representación, para realizar el cálculo de la posición de un componente se ha generalizado la ecuación de forma que se consideren todos los componentes contenedores en que pueda estar incluido el componente al que se le quiere calcular su posición en el dispositivo de visualización

$$Pos_{B_{cd}}(x) = Pos_{B_{cc}}(x) + \sum Pos_{Comp_{cs}}(x)$$
$$Pos_{B_{cd}}(y) = Pos_{B_{cc}}(y) + \sum Pos_{Comp_{cs}}(y)$$

donde:
cd=coordenadas de dispositivo,
cc=coordenadas componente y
cs=coordenadas componente contenedor superior

El sumatorio se realiza sobre las posiciones de los sucesivos componentes contenedores en los que esté contenido, hasta llegar al componente contenedor independiente.

La formulación general para el cálculo de la posición, en el caso de una modificación de posición de un componente en el dispositivo de visualización, corresponde con la transformación de traslación en geometría. Para esta transformación existen dos

ecuaciones que definen cómo realizar la correspondencia entre un punto (X, Y) en un punto transformado (X', Y')[Ols98].

$$X' = X + dx$$
$$Y' = Y + dy$$

Los coeficientes a aplicar a las ecuaciones para controlar la transformación son dx y dy. El conjunto de posibles valores para los coeficientes determina el conjunto de posibles transformaciones. En nuestro ejemplo los coeficientes son $dx = 90$ y $dy = 90$.

4.3.2 Validez de la propiedad de tamaño

Para verificar la validez de la propiedad de tamaño, debemos comprobar que las propiedades definidas para la representación de una interfaz se mantienen aunque se realicen modificaciones en el tamaño de los componentes contenedores. Las modificaciones sólo se permiten si las transformaciones de tamaño sobre los componentes representados no afectan a sus propiedades.

Como verificación inicial de transformación de tamaño o escalado vamos a ver cómo se realiza una operación visual de cambio de tamaño de un componente contenedor y cómo ésta se mantiene en sus componentes contenido. Sea un componente contenedor A que se redimensiona a un tamaño diferente, el cual denotaremos como A'. Se define una función de cálculo de tamaño para los componentes contenido, dada una operación de redimensionamiento f_{TAM} del componente contenedor.

$$f_{TAM}(B')(x) = \frac{(Tam_B(x) Tam_{A'}(x)}{Tam_A(x)}$$
$$f_{TAM}(B')(y) = \frac{(Tam_B(y) Tam_{A'}(y)}{Tam_A(y)}$$

donde

$Tam_{componente}(x)$ es el tamaño en el eje X de un componente
B' es el componente B con el tamaño resultante después de la operación de redimensionado del componente contenedor.

Estas ecuaciones corresponden con la transformación de escala habitual:

$$X' = X.sx$$
$$Y' = Y.sy$$

de donde

$$sx = \frac{X'}{X}$$
$$sy = \frac{Y'}{Y}$$

donde sx y sy son, en nuestra representación, números positivos (puesto que a nivel de interfaz de usuario no permitimos reflexiones sobre ejes de componentes gráficos) que permiten en coeficiente superiores a 1 aumentar el tamaño, y en coeficientes entre 0.0 y 1.0 disminuir el tamaño. $f_{TAM}(A')$ nos proporciona el nuevo tamaño A' de A tras una transformación de tamaño.

Supongamos una operación de modificación de tamaño sobre el componente A. Si su nuevo tamaño es (200,200), las operaciones para el cálculo de tamaño sobre el componente B serían:

$$f_{TAM}B(x) = Round(\frac{(50)(200)}{130} = 76,9) \Rightarrow 77$$
$$f_{TAM}B(y) = Round(\frac{(110)(200)}{130} = 169,2) \Rightarrow 169$$

Dado que una modificación de tamaño afecta a las propiedades de posición relativas topológicas (centrado, alineado, justificado y equiespaciado), debe comprobarse que una vez realizado el cálculo del nuevo tamaño de los componentes contenedores pueden satisfacerse las propiedades de posición relativa topológica. Como se ha visto anteriormente, una función de transformación de tamaño nos permite calcular automáticamente el tamaño de los componentes contenido a partir de la modificación de tamaño de un componente contenedor. Si encontramos una función que permita calcular la posición relativa topológica de los componentes podemos definir el siguiente diagrama conmutativo:

$$\begin{array}{ccc} \text{Repr. Abst. A,B} & \xrightarrow{f_{Centrado}} & \text{Repr. composición A,B} \\ \downarrow f_{Tam} & & \downarrow f_{Tam} \\ \text{Repr. Abst. A',B'} & \xrightarrow{f_{Centrado}} & \text{Repr. composición A',B'} \end{array}$$

Supongamos la propiedad de posición relativa Centrado. Esta podría obtenerse de la siguiente manera, dado un componente contenedor A y un componente contenido B:

$$f_{centrado}(B)(x) = Round((Tam_A(x) - Tam_B(x))/2)$$
$$f_{centrado}(B)(y) = Round((Tam_A(y) - Tam_B(y))/2)$$

Podemos generalizar esta notación y expresarla como $f_{Centrado}(B)$, que nos devuelve la posición de B centrado en su componente contenedor.

Se ha demostrado que tras una operación visual de redimensionado se puede obtener el nuevo tamaño de todos los componentes implicados. De la misma forma se ha demostrado la aplicación de una función de posición sobre un tamaño de componente. Esta función de posición puede aplicarse sobre el tamaño de los componentes previo al redimensionamiento o posterior al redimensionamiento. De la misma forma, la función de tamaño puede aplicarse sobre la representación abstracta de los componentes sin restricción de posición o bien con restricción de posición.

Una vez definidas la función de posición centrado y la función de tamaño, se verifica la existencia de todas las funciones implicadas en el diagrama conmutativo y el mantenimiento de la propiedad de posición relativa centrado para el nuevo tamaño.

$$\begin{array}{ccc} \text{Repr. Abst. A,B} & \xrightarrow{\exists Pos_{Centrado}(A,B)} & \text{Repr. composición A,B} \\ \downarrow \exists Tam(A,A')(B) & & \downarrow \exists Tam(A,A')(B) \\ \text{Repr. Abst. A',B'} & \xrightarrow{\exists Pos_{Centrado}(A',B')} & \text{Repr. composición A',B'} \end{array}$$

La verificación para el resto de las propiedades relativas de posición (justificado, alineado y equiespaciado) es similar a la anterior y trivial una vez definida la función de posicionamiento correspondiente.

4.4 Conexión con la aplicación (Entrada, Salida y Procesos)

En esta sección se introducen los elementos de la representación que permiten la definición de acciones de introducción y muestra de datos desde la funcionalidad de la interfaz, y la invocación de procesos previsibles en la aplicación subyacente.

Con el objeto de representar la posibilidad de que el usuario realice la acción de introducir información, o la visualización de información hacia el usuario, se han introducido dos elementos en la representación de los componentes: InfI(Id)(T/F) e InfO(Id)(T/F). Los Id son identificadores que se asignan a los elementos InfI e InfO que permiten identificar informaciones diferentes.

El elemento InfI(Id) puede tomar dos valores: T ó F. Si se asigna el valor T significa que el usuario puede introducir información en el componente que tiene InfI(Id). En el caso de la asignación del valor F al elemento InfI(Id) el usuario no puede introducir información.

Si un InfO(Id) toma el valor T, significa que se muestra a través del mismo información al usuario; en caso de valor F permanece con la información que poseía.

Estos elementos de representación no son realmente componentes de la interfaz sino que se corresponden con funciones de entrada/salida, puesto que no se define para ellos ninguna apariencia. La apariencia de estos elementos viene dada por la de los elementos que los contienen, como se describe en la representación de la composición de la interfaz. Por su naturaleza funcional son un caso atípico dentro de la representación, aparecen como propiedades de los componentes que los contienen y pueden tener dos valores de funcionamiento (T y F).

Asociado a la representación del diálogo de componentes de la interfaz está la invocación de procesos previsibles de la aplicación, que el diseñador intuye como necesarios para la devolución de una respuesta por parte de la aplicación hacia el usuario. Para representar esta invocación se utiliza Proc(Id), donde ID es un identificador que se usa con el fin de distinguir procesos entre sí, o bien identificar el mismo proceso invocado en más de una acción del usuario.

Una ventaja que incorpora la introducción de estos elementos dentro de la representación es que permite un análisis de los requerimientos de la aplicación a nivel de información de entrada y salida, además de servir de guía para el primer análisis de la aplicación al identificar procesos que van a ser necesarios en la aplicación.

4.5 Diálogo entre los componentes de la representación

Para la descripción del diálogo se incluye en la representación de componentes dos propiedades:

- Activo: su valor puede ser T (true=cierto) o F(false=falso). Cuando vale T indica que el componente puede responder a eventos asociados

- Visible: igual que en el caso anterior, su valor puede ser T (true=cierto) o F(false=falso). El valor T determina que el componente posee apariencia visual perceptible por el usuario, aunque otro componente lo oculte totalmente. En este caso, el usuario no podrá percibirlo aunque el valor de Visible sea T. Es similar a lo que ocurre en una representación 3D donde un objeto tridimensional está incluido dentro de otro objeto opaco. El usuario no puede ver el objeto del interior aunque éste tenga forma, color y textura. El motivo de la existencia de esta propiedad se explicará en detalle al tratar el concepto de jerarquía visual.

El valor de la propiedad Activo(T/F) es diferente si el componente es un componente contenedor. En el caso de que un componente contenedor posea la propiedad Activo(F) significa que aunque alguno de sus componentes contenido posea la propiedad Activo(T) no responderá a sus acciones asociadas. Esto es muy útil para desactivar en su totalidad un componente contenedor simplemente colocando el valor F en su propiedad Activo, mientras que sus componentes contenidos siguen manteniendo los valores correspondientes en su propiedad Activo. Al poder desactivar de forma general un componente contenedor se conserva su estado de diálogo, lo cual no se podría lograr si se pusiesen a valor F en la propiedad Activo todos los componentes contenido.

La representación de los componentes de la interfaz es fija para la representación de la interfaz. Los aspectos dinámicos producidos por la interacción del usuario con la interfaz están reflejados en la representación del diálogo de los componentes.

Para representar el diálogo entre componentes dentro de la interfaz de usuario se utiliza la notación correspondiente la tabla B.2. En la representación del diálogo se asocia a cada componente el evento al cual responde junto con los cambios que provoca en la interfaz.

Dentro de las posibles respuestas se encuentran la activación o desactivación de otro componente con el objeto de que pueda o no responder a un evento, y la activación o desactivación de la propiedad Visible para que tenga o no apariencia perceptible por el usuario.

Podemos encontrar tres casos especiales de respuesta:

- activación/desactivación de una función de entrada de información, identificada por un ID

Item	Tipo	Descripción
DIAG:	Ob	Cadena identificación comienzo diálogo componente
<nombre de componente>	Ob	Nombre del componente que tiene un diálogo asociado en la interfaz
#	Ob	Separador de componente y el evento asociado
<Evento>	Ob	Evento asociado al componente
<Evento> LeftClick RighClick ReLeClick ReRiClick MouseOn Key(tecla/s) KeyPress(tecla/s) KeyRele(tecla/s)		Opciones en Evento click botón izquierdo click botón derecho Soltar el botón izquierdo Soltar el botón derecho Puntero del ratón sobre el componente Tecla pulsada y soltada Tecla pulsada y retenida Tecla soltada
#	Ob	Separador del componente y el evento asociado
<Respuesta>	Ob	Respuesta del componente dentro de la interfaz ante la ocurrencia del evento
<Respuesta> <Nombre de componente> Activo(T/F) Visible(T/F) InfI()(T/F) InfO()(T/F) Proc(Id)		Opciones en <Respuesta> Nombre del componente que responde Activación o desactivación de un componente Propiedad de visible o no de un componente Función de Entrada Salida Proceso de la aplicación que se invoca ante el evento sobre el componente. La identificación del proceso se realiza a través del Id

Tabla 4.10: Formato del fichero de diálogo de los componentes de la interfaz

- activación/desactivación de una función de salida de información, identificada por un ID
- invocación de un proceso identificado por un ID y conocido por la aplicación

A nivel de eventos posibles en el diálogo de componentes se ha identificado un conjunto mínimo de acciones que el usuario puede llevar a cabo en una interfaz de usuario de manipulación directa. Cualquier acción del usuario puede realizarse a partir de los eventos del conjunto aquí definido. La representación admite la inclusión de nuevos eventos asociados a acciones de usuario que puedan ser producidas por dispositivos de interacción distintos de los aquí contemplados (ratón y dispositivo apuntador).

Mediante el formato de descripción del diálogo, junto con el formato de descripción de componente y el de descripción de composición, es posible representar la apariencia y comportamiento de la interfaz global. En la representación de los componentes se establecen las propiedades iniciales de Visible y Activo.

A partir de los valores de las propiedades Activo y Visible de la descripción de los componentes puede establecerse un primer estado de la interfaz de usuario. Se entiende como estado de la interfaz, aquella combinación de componentes de la interfaz en la cual se espera por una acción del usuario. El estado inicial de la interfaz viene dado por la combinación de todos los componentes que poseen la propiedad Visible=T, Activo=T y los eventos asociados. La lista de eventos asociados con los componentes y sus respuestas se encuentran en la descripción del diálogo.

$$E_x = C_0, C_1, ..., C_m \forall C_i \# \text{Visible(T) y/o Activo(T)}$$

4.5.1 Determinación de estados de la interfaz

Desde el estado inicial puede obtenerse el conjunto de estados posibles siguientes, mediante la aplicación de cada uno de los eventos posibles sobre los componentes activos del estado inicial. A partir de los estados generados, y aplicando el mismo proceso, pueden obtenerse los estados siguientes y llegar hasta un estado final en donde ningún componente de la interfaz tenga la propiedad Activo=F y Visible=F.

Durante el proceso de obtención de estados de la interfaz de usuario pueden determinarse transiciones mediante los eventos provocados por el usuario hacia estados ya existentes.

Dos estados son iguales (y, por lo tanto, el mismo estado) si todos sus componentes poseen valores iguales en las propiedades Activo y Visible. Los componentes que forman parte de un estado son los que tienen una funcionalidad dentro del estado. Esta funcionalidad puede venir dada por una apariencia visual que proporciona al usuario información relevante dentro de la interfaz, o por una respuesta asociada a un evento del usuario sobre una zona determinada de la pantalla o sobre alguno de sus componentes contenido en el caso de un componente contenedor. La modificación de propiedades visuales sobre componentes integrantes de un estado de la interfaz, no modifica su funcionalidad dentro de la interfaz. Una de las causas posibles para la modificación de la apariencia de un componente dentro de la interfaz es una preferencia por parte del usuario o una adecuación al contexto de apariencia del estado.

Si la modificación de la apariencia del componentes es tal que su funcionalidad varía, estamos en el caso de un componente distinto. Esta situación sobre los modelos que utilizan interadores para la especificación de componentes corresponde con la de los distintos estados que un componente puede alcanzar a través de las acciones del usuario. Según la especificación tradicional de interador para cada estado del interador existe una funcionalidad distinta y una única apariencia para el interador. En ningún modelo se contemplan múltiples funciones de visualización para un único estado del interador. Ello es debido a que la especificación se centra en el diálogo del interador en vez de en su apariencia visual.

En esta propuesta, que parte de la apariencia de los componentes en la interfaz, pueden existir distintas apariencias para un componente, siempre provocadas por acciones de usuario (utilizando operaciones visuales), pero sin embargo la funcionalidad del componente sigue siendo la misma. Utilizando esta propuesta se mantiene de forma implícita la consistencia visual de la interfaz, puesto que dos componentes que se crean con la misma apariencia deberían tener la misma funcionalidad. Si el usuario modifica la apariencia de un componente lo hace como una elección personal puesto que esa modificación no afectará a su funcionalidad.

En la propuesta no se contempla la representación abstracta de información de aplicación (parte de la interfaz correspondiente al área de trabajo de la interfaz [RGP00]) dado que esta no puede realizarse en un sistema de prototipado en las primeras fases del desarrollo de una aplicación. Si una acción del usuario sobre la interfaz cambiase la apariencia de la misma de una forma drástica, esta apariencia correspondería con un estado de la interfaz diferente. Un ejemplo de esto sería la elección en una interfaz

de una presentación determinada siguiendo una guía de estílo diferente. No se modifican las propiedades Visible ni Activo de los componentes, pero ya no serían los mismos componentes. Volviendo a la relación con los modelos de interador, puesto que esta propuesta parte de la apariencia visual de los componentes, está claro que los componentes serían otros, y en el caso de un iterador, puesto que no existe forma de realizar una función diferente para el mismo estado, tendría que ser un estado distinto.

Si la interfaz es correcta debería existir sólo un estado en el cual todos los componentes tienen las propiedades Activo=F y Visible=F (estado final de la interfaz). Igualmente podemos identificar de forma inequívoca el estado inicial de la interfaz de usuario a partir de la representación de los componentes de la interfaz.

La representación de los componentes de la interfaz, la composición de la interfaz, y el diálogo entre componentes es constante para una interfaz de usuario. Por lo tanto, para la descripción de los estados de la interfaz se ha definido un formato de representación de los mismos que se representa en la tabla B.9. La información representada para cada uno de los estados corresponde con los valores de las propiedades de los componentes de la interfaz.

Para representar la transición entre estados, identificando las acciones de usuario que provocan dichas transiciones, se utiliza el formato correspondiente a la tabla B.10 donde se indica un estado, un evento, el componente sobre el cual se aplica el evento, y el nuevo estado que se provoca.

Para ilustrar la generación de estados, se presenta el siguiente ejemplo:

Información de entrada: Disponemos de las siguientes descripciones:

D1: Descripción de los componentes de la interfaz

D2: Descripción de la composición de la interfaz

D3: Descripción del diálogo entre componentes.

Información de salida: A partir de las descripciones anteriores, deseamos obtener:

D4: Conjunto de estados posibles de la interfaz.

D5: Conjunto de transiciones entre estados (acciones de usuario).

Supongamos una interfaz de usuario formada por los siguientes componentes:

C el componente contenedor de la interfaz

D, E, A, B, F y salir son componentes contenido para la visualización de datos

El evento LeftClick sobre el componente A provoca la aparición del componente D

El evento LeftClick sobre el componente B provoca la aparición del componente E

El evento LeftClick sobre el componente F provoca la desaparición de los componentes D y E

El evento LeftClick sobre el componente salir provoca la desaparición y desactivación del componente C.

Contenido de los ficheros de entrada (Con objeto de simplificar la notación, sólo se han incluido en las representaciones la información relevante a este proceso):

D1: Fichero de descripción de componentes

COMP:C#R(100,100)#Activo(T)#Visible(T)

COMP:A#R(10,10)#Activo(T)#Visible(T)

COMP:B#R(10,10)#Activo(T)# Visible(T)

COMP:salir#R(10,10)#Activo(T)#Visible(T)

COMP:D#R(10,80)#Activo(F)#Visible(F)

COMP:E#R(10,80)#Activo(F)#Visible(F)

COMP:F#R(10,10)#Activo(F)#Visible(F)

De lo que se deduce que, en el estado inicial, los componentes activos son C, A, B y salir.

D2: Fichero de contenido: Descripción de la composición de la interfaz

COMP:C#(A#B#salir#D#E#F)

D3: Fichero de diálogo entre componentes

DIAG:A#LeftClick#DVisible(T)

DIAG:A#LeftClick#FVisible(T)Activo(T)

DIAG:B#LeftClick#EVisible(T)

DIAG:B#LeftClick#FVisible(T)Activo(T)

DIAG:salir#LeftClick#CVisible(F)Activo(F)

DIAG:F#LeftClick#DVisible(F)

DIAG:F#LeftClick#EVisible(F)

Obtención de D4 y D5:

Obtención de la información de salida en el ejemplo: La interfaz comienza en un estado inicial E_0 cuya definición es la siguiente:

ESTA:0#
COMP:C#Activo(T)#Visible(T)
COMP:A#Activo(T)#Visible(T)
COMP:B#Activo(T)#Visible(T)
COMP:salir#Activo(T)#Visible(T)

Analizamos si el componente B posee algún evento asociado en D3. Aparecen dos entradas correspondientes al evento LeftClick, que provoca que el componente E y el componente F transformen su propiedad Visible a T. Además, el componente F transforma su propiedad Activo a T. El nuevo estado tiene la siguiente representación:

ESTA:1#
COMP:C#Activo(T)#Visible(T)
COMP:A#Activo(T)#Visible(T)
COMP:B#Activo(T)#Visible(T)
COMP:F#Activo(T)#Visible(T)
COMP:salir#Activo(T)#Visible(T)
COMP:E#Activo(F)#Visible(T)

Se le ha asignado el valor 1 como identificador tras comprobar si los componentes del nuevo estado y sus propiedades son iguales a alguno de los estados ya identificados y ver que no es así (solo tenemos identificado el estado 0). Ahora se añade en la representación de transiciones la entrada:0#B#LeftClick#1

Siguiendo el mismo procedimiento, y mientras no agotemos los componentes activos y con evento asociado, buscamos el siguiente estado de E_0. Asociado al componente A está el evento LeftClick que provoca, para los componentes D y F, el valor de la propiedad Visible a T. Además, provoca la transformación de la propiedad Activo de F a T. Se comprueba que este estado no está definido, se le asigna un nombre de estado nuevo (el 2), se representa su composición y su transición.

0#A#LeftClick#2

ESTA:2#
COMP:C#Activo(T)#Visible(T)
COMP:A#Activo(T)#Visible(T)
COMP:B#Activo(T)#Visible(T)
COMP:F#Activo(T)#Visible(T)
COMP:salir#Activo(T)#Visible(T)
COMP:D#Activo(F)#Visible(T)

El componente salir con el evento LeftClick provoca un nuevo estado correspondiente a la transformación de las propiedades Visible y Activo del componente C a F. Este estado conlleva que al estar en valor F las propiedades Activo y Visible del componente contenedor principal de la interfaz implica la destrucción de la interfaz por ser el contenedor principal.

0#salir#LeftClick#3

ESTA: 3#
COMP:C#Activo(F)#Visible(F)

Como ya no existe ningún componente activo en el estado E_0 que no haya sido procesado, se sigue el mismo procedimiento con el siguiente estado no procesado.

Sobre E_1 el componente B tiene asociado el evento LeftClick. Una vez representados los cambios se observa que el estado ya existe y coincide con E1. En este caso, no se genera un nuevo estado sino únicamente se indica la transición a un estado ya existente:

1#B#LeftClick#1

Siguiendo con el mismo proceso, se identifican los siguientes estados y transiciones:

1#A#LeftClick#4

ESTA:4#
COMP:C#Activo(T)#Visible(T)
COMP:A#Activo(T)#Visible(T)
COMP:B#Activo(T)#Visible(T)
COMP:F#Activo(T)#Visible(T)
COMP:salir#Activo(T)#Visible(T)
COMP:D#Activo(F)#Visible(T)
COMP:E#Activo(F)#Visible(T)

1#salir#LeftClick#3
1#F#LeftClick#5

ESTA:5#
COMP:C#Activo(T)#Visible(T)
COMP:A#Activo(T)#Visible(T)
COMP:B#Activo(T)#Visible(T)
COMP:F#Activo(T)#Visible(T)
COMP:salir#Activo(T)#Visible(T)

2#B#LeftClick#4
2#A#LeftClick#2
2#salir#LeftClick#3
2#F#LeftClick#5
4#B#LeftClick#4
4#A#LeftClick#4
4#salir#LeftClick#3
4#F#LeftClick#5
5#B#LeftClick#1
5#A#LeftClick#2
5#salir#LeftClick#3
5#F#LeftClick#5

Puede verse una visualización de los estados en la figura 4.7.

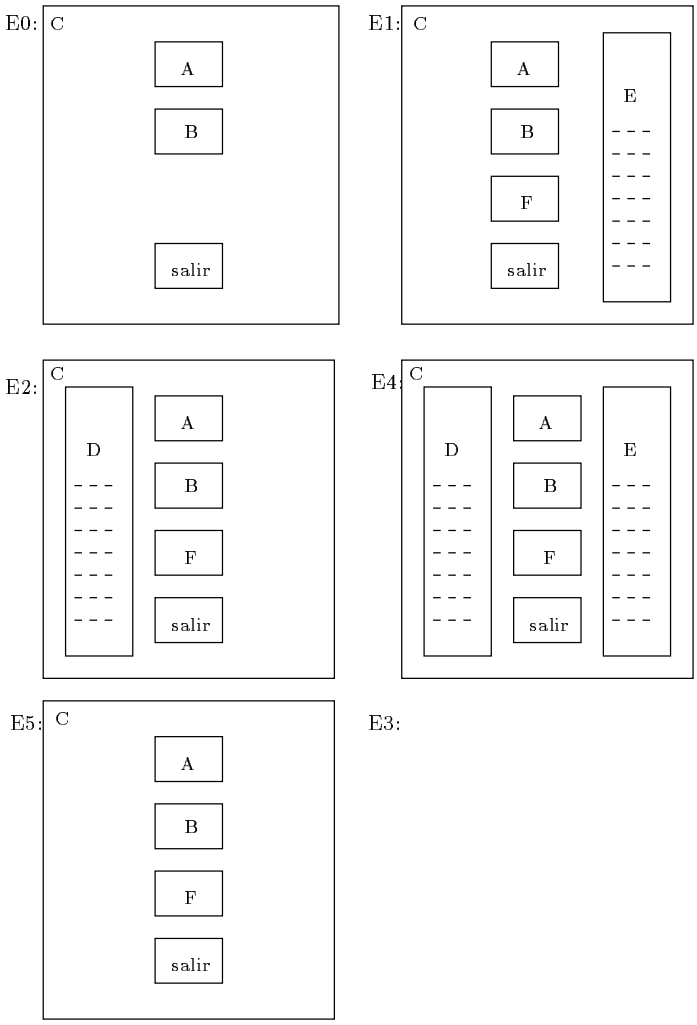

Figura 4.7: Visualización de los estados del ejemplo propuesto

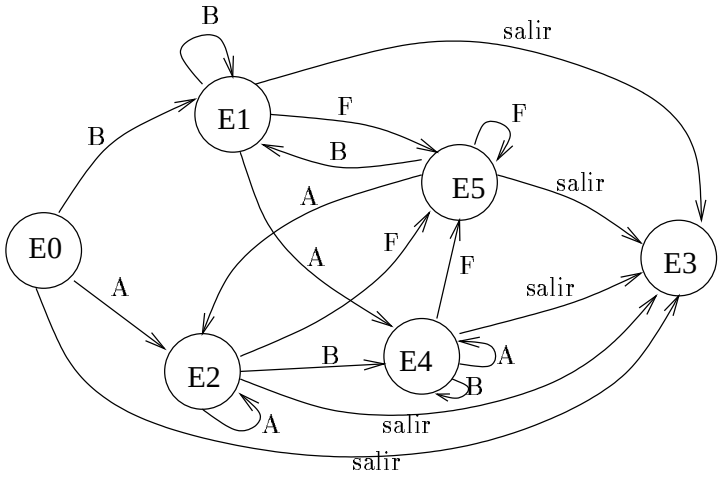

Figura 4.8: Grafo de transición entre estados

4.5.2 Grafo de estados de la interfaz de usuario

Como elemento gráfico para representar la transición entre estados vamos a utilizar un grafo dirigido, donde los nodos corresponden a los estados de la interfaz y los arcos a las transiciones entre estados. Los arcos se etiquetan con el componente y el evento que provoca la transición. En el presente ejemplo, debido a que el evento de transición es siempre LeftClick no se ha representado (Figura 4.8).

En el grafo de estados de la interfaz puede observarse la existencia de dos estados particulares. Corresponden a dos nodos, en los cuales la ocurrencia singular de arcos de entrada o salida los convierten en estados determinantes para la funcionalidad de la interfaz.

Un nodo en el cual todos sus arcos asociados son de salida y no tiene ningún arco de entrada desde un nodo al cual no se pueda llegar sin pasar por él mismo, representa el estado inicial de la interfaz. Un nodo en el cual todos sus arcos asociados son de entrada y ninguno de salida, representa el estado final de la interfaz. Este estado final es el asociado al valor F en las propiedades Activo y Visible del componente contenedor principal de la interfaz. La existencia de dos o más nodos que tengan asociados únicamente arcos de entrada implica una situación anómala puesto que corresponde con dos estados finales diferentes.

4.6 Jerarquía de dependencia y visual

Hasta el momento se ha establecido la representación de la apariencia de los componentes de la interfaz y del diálogo que se produce entre ellos. A partir de esta representación pueden obtenerse los estados de la interfaz, tomando como estado cada uno de los momentos de interacción en que el sistema aguarda por una acción de usuario.

Es adecuado proporcionar un sistema de representación que permita observar gráficamente la apariencia y el comportamiento de los componentes desde un punto de vista más general. Para ello vamos a representar, por un lado, los componentes indicando su grado de dependencia, y por otro lado las acciones que el usuario puede llevar a cabo sobre ellos y su respuesta correspondiente. Mediante la representación de estas dos jerarquías se proporciona un elemento gráfico que permite identificar rápidamente los componentes que forman parte de las capas visuales implicadas en la interfaz.

La razón del estudio de las capas visuales viene dada por dos motivaciones principalmente:

- Las interfaces de manipulación directa existen sobre arquitecturas software basadas en sistemas de ventanas, en donde la información se organiza en ventanas que utilizan dimensiones y media, como se ha comentado anteriormente. Hasta este momento se ha evitado la referencia a cualquier elemento identificable como estándar dentro de las interfaces de usuario actuales. Sin embargo, es necesario en este punto centrar la necesidad de esta representación dado que no existen estudios que permitan estudiar la influencia de las ventanas en las interfaces actuales.

- Los sistemas de ventanas utilizan la metáfora de objetos apilados. Este apilado de objetos se realiza sin profundidad, de ahí la media dimensión, y por ello no se consideran las interfaces actuales interfaces en tres dimensiones. La metáfora de objetos apilados hace comprensible al usuario que el elemento que está en la parte superior de la pila es aquel sobre el cual se pueden centrar sus acciones. Aquí se identifica otra intervención del factor humano de percepción sobre la interfaz de usuario, el efecto de apilado. Corresponde con un ajuste óptico que el usuario realiza sobre los componentes al asumir que el objeto que tapa o se superpone sobre otro está más cercano.

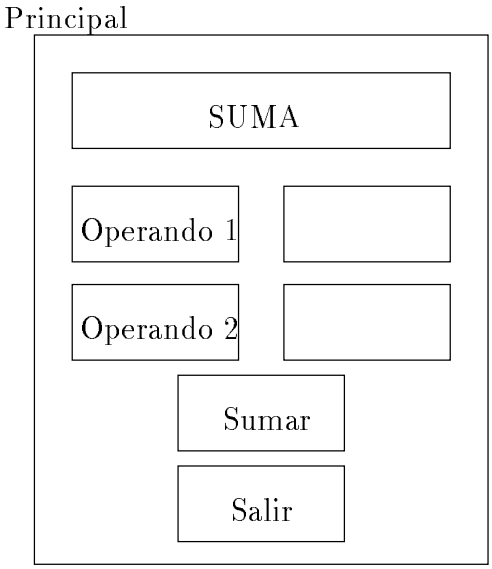

Figura 4.9: Ejemplo de interfaz de un componente contenedor

Si para un determinado estado de la interfaz existen varios componentes contenedores independientes pueden darse varias jerarquías de capas para el mismo estado. La ocurrencia ordenada de los componentes contenedores independientes, a lo largo de la interacción del usuario con la interfaz, determina la colocación en un nivel u otro de la jerarquía de los componentes contenedores, y por lo tanto su visibilidad y alcanzabilidad. Esto se verá más en detalle en el análisis de la interfaz.

Dada la descripción de la composición de la interfaz, podemos obtener cuál es el componente de nivel más alto, aquel que además de no estar contenido en ningún otro forma parte del estado inicial de la interfaz. A partir de este componente crearemos una jerarquía de dependencia, incluyendo todos aquellos componentes que estén contenidos en él. La interfaz de la figura 4.9 determina la jerarquía de dependencia mostrada en la figura 4.10.

Se representa el componente contenedor y se une con arcos continuos a los componentes contenido que posee, independientemente de sus propiedades de activación y visibilidad.

Definición Denominamos jerarquía de dependencia la que se establece por dependencia de existencia entre los componentes visuales abstractos de la interfaz.

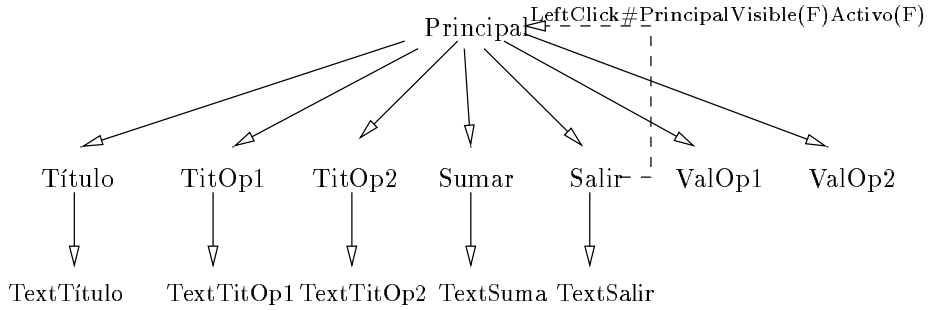

Figura 4.10: Representación de jerarquía de dependencia de un nivel

Las acciones del usuario se representan mediante arcos discontinuos que unen el componente sobre el cual se aplica el evento con el componente que recibe el cambio en la interfaz provocado por el evento. Un arco de acción de usuario tiene asociado una etiqueta en la cual se indica el evento que lo provoca.

Cada componente contenedor posee su propia de jerarquía de dependencia. Esta jerarquía será independiente solo si el componente contenedor no es a su vez componente contenido de otro componente contenedor. Los componentes contenedores independientes se relacionan en la interfaz de usuario a través de las acciones del usuario. Todos los componentes contenido que forman parte de un componente contenedor se caracterizan por tener un significado o propósito común. Si el componente contenedor no es visible ni activo, y por lo tanto no accesible para el usuario en el momento de comenzar a trabajar con la interfaz, es porque una elección del usuario sobre la interfaz provocará su aparición y uso.

Cuando una acción del usuario activa o pone visible un componente que está contenido en el mismo componente contenedor del componente que recibe el evento, se representa esta acción con un arco discontinuo horizontal, de forma que se indica que pertenece al mismo nivel de jerarquía visual.

Cuando la acción del usuario activa o pone visible un componente contenedor que es independiente de aquel que recibe la acción del usuario, se representa con un arco discontinuo vertical hacia abajo, y el nuevo componente contenedor se coloca por debajo del anterior. Con ello se representa la activación y aparición más tardía del componente contenedor. Cuando un componente contenedor se representa un nivel más bajo que otro significa que su aparición se realizará sobre ese otro componente contenedor. Visualmente y a nivel de acciones de usuario estará más al alcance del

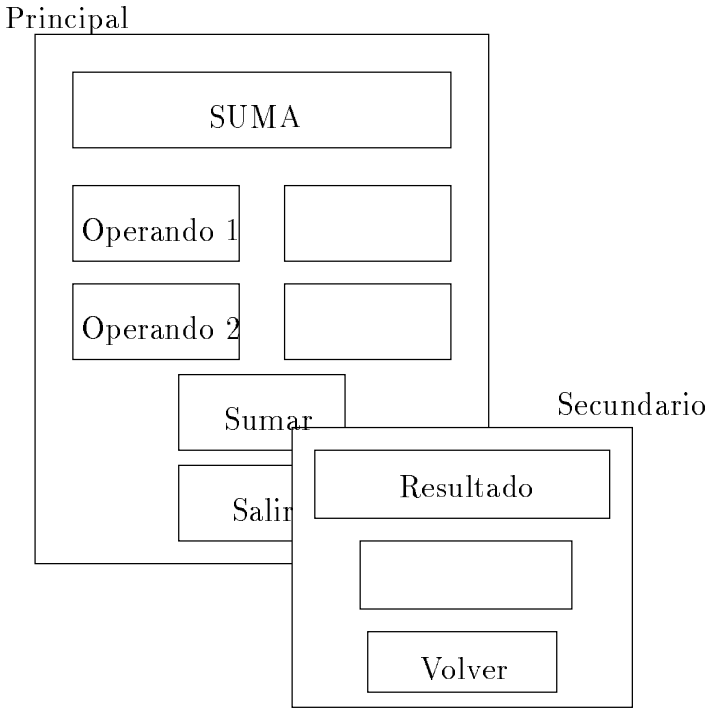

Figura 4.11: Ejemplo de interfaz con dos componentes contenedores

usuario (Figura 4.11).

Definición Denominamos jerarquía visual a la que se establece entre los componentes visuales abstractos de la interfaz debida a la ejecución ordenada temporalmente de acciones de usuario.

En el capítulo dos se ha introducido el concepto de interfaz de usuario multimodal relacionado con las interfaces de usuario de manipulación directa. Se utilizará el concepto *modal* a partir de aquí para indicar un componente contenedor que en un momento dado puede recibir todas las acciones del usuario, estableciendo la propiedad Activo(F) del resto de componentes contenedores que forman parte de ese estado. Se utilizará el concepto *interaccionable* para referirnos a un componente contenedor al cual puede aplicársele una acción de usuario de entre las permitidas. El componente contenedor interaccionable/no interaccionable tiene una gran importancia en la jerarquía visual dentro de las interfaces de usuario estáticas. Ello es debido a que, en el caso de una interfaz de usuario estática, la existencia de un componente no interaccionable que visualmente impida el acceso por parte del usuario a una acción sobre un

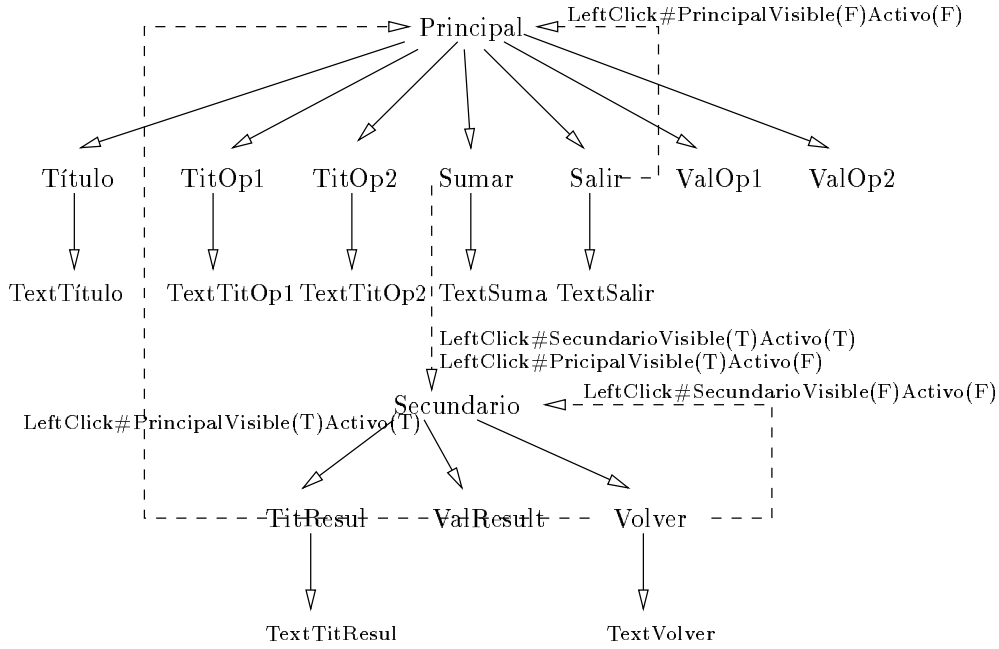

Figura 4.12: Representación de jerarquía visual de dos niveles

componente interaccionable del nivel jerárquico inferior, inhabilita dicha acción del usuario como parte del conjunto de comandos que forman parte de las tareas identificadas en la sección anterior. En el caso de una interfaz dinámica, si la propiedad de posición del componente de nivel jerárquico superior es relativa, puede accederse al nivel jerárquico inferior a través de las operaciones visuales (cambio de posición del componente contenedor). En el caso de que la propiedad de posición sea fija, encontramos el mismo problema que en el caso estático.

CONSECUENCIA: LA APARIENCIA VISUAL Y LA COLOCACIÓN DE COMPONENTES EN LA INTERFAZ DE USUARIO PUEDEN AFECTAR AL CONJUNTO DE TAREAS QUE EL USUARIO PUEDE REALIZAR SOBRE LA MISMA.

El estudio de la problemática de la jerarquía visual de componentes puede resumirse de la siguiente manera:

Para todo componente contenedor C perteneciente al estado E_i y con propiedad Activo(T), y para todo componente contenedor C' perteneciente al estado E_{i+1} con propiedad Visible(T), siendo C y C' componentes contenedores independientes entre sí, si la intersección del Area(C) y Area(C') es distinto del conjunto vacío se produce un conflicto.

Este conflicto puede tener diferentes grados de importancia:

- En el caso de que la activación del componente C' tenga asociada la desactivación del componente C (aunque no su visibilidad) no existe ningún problema excepto la necesidad de un componente contenido de C' que tenga asociado un evento que permita la desactivación de C'. En caso contrario tendríamos un estado final de interfaz (no tiene arcos hacia el resto de estados de la interfaz) que no correspondería con la propiedad Activo(F) y Visible(F) del componente principal de la interfaz.
- En el caso de que la activación del componente C' no tenga asociada la desactivación del componente C, nos encontramos con una jerarquía en donde los dos componentes contenedores son interaccionables. La problemática es diferente en el caso de una interfaz estática que dinámica.
  - En el caso de una interfaz dinámica, si la propiedad de posición de C' es relativa, puede accederse al componente C utilizando las operaciones visuales (modificación de la posición de C').
  - En el caso de una interfaz estática, o dinámica con posición fija para el componente C', si la intersección entre el Area(C) y el Area(C') es igual al Area(C') no se puede acceder al componente C aunque tenga sus propiedades Activo(T) y Visible(T). Si la intersección entre el área de los dos componente es distinta del Area(C') puede accederse al componente C, aunque el usuario, dependiendo de la proporción de Area(C) que no corresponde con Area(C'), puede tener problemas para identificar los componentes contenido de componente C.

En el caso de que el componente C' que se activa y ocupa el nivel jerárquico superior sea un componente dependiente de C, el problema es similar al de componentes contenedores independientes y se resuelve del mismo modo. La definición de un componente contenido que ocupe una capa jerárquica superior a la del componente contenedor en el cual está incluido, atenderá generalmente a un motivo muy específico y temporalmente muy reducido sobre un estado concreto de la interfaz.

4.7 Conclusiones del capítulo

La importancia de la apariencia visual no está recogida en los modelos ni en las herramientas actuales. Por lo tanto, es necesaria una representación abstracta de la misma para poder estudiar su participación en la interacción del usuario con la interfaz.

En la mayoría de los casos, a la hora de construir una interfaz de usuario, se utilizan widgets estándar comerciales, los cuales limitan la creatividad de los diseñadores gráficos cuando tienen que construir las interfaces de usuario. Con la representación abstracta propuesta, el diseñador es libre para decidir la forma y comportamiento de los elementos que componen la interfaz de usuario. Por ello, el diseño basado en la componente visual puede tener ventajas sobre los modelos y herramientas actuales.

La representación abstracta de la composición entre componentes proporciona un método de definición de la colocación de componentes, utilizando restricciones topológicas que no implican necesariamente relaciones entre componentes del mismo nivel. Las relaciones se establecen de forma jerárquica, con lo cual las propiedades de posición y tamaño existentes involucran un único componente contenedor. La ventaja de este sistema es la garantía de una solución única para la colocación de los componentes en el dispositivo de visualización a partir de la representación abstracta de la composición. Otra ventaja de este sistema es la no necesidad de resolutores (solver) de restricciones geométricas para obtener el resultado final sobre el sistema de visualización.

La representación abstracta de la interfaz también permite estudiar un aspecto hasta ahora no contemplado en ninguna herramienta: la influencia de la jerarquía de capas visuales en el diseño de la interfaz de usuario, su influencia en la percepción de las acciones del usuario (visibilidad) y en la capacidad de llevarlas a cabo (alcanzabilidad).

Los sistemas de ventanas utilizan la metáfora de objetos apilados. Este apilado de objetos se realiza sin profundidad, de ahí la media dimensión y por ello no se consideran las interfaces actuales interfaces en tres dimensiones. La metáfora de objetos apilados hace comprensible al usuario que el elemento que está en la parte superior de la pila es aquel sobre el cual se pueden centrar sus acciones. Aquí se identifica otra intervención del factor humano de percepción sobre la interfaz de usuario, el efecto de apilado corresponde con un ajuste óptico que el usuario realiza sobre los componentes

al asumir que el objeto que tapa o se superpone sobre otro está más cercano. Con la jerarquía de capas se puede representar el apilado de componentes y mediante el mismo pueden hacerse estudios sobre:

- La visibilidad. Identificando el número de solapamientos que existe para un estado y cual es el área visible por el usuario de entre las capas solapadas.
- La complejidad de la interfaz. Identificando el número de capas con componentes con acciones de usuario asociadas para un estado.
- La comprensión de la interfaz por parte del usuario. Identificando los comandos que el usuario tiene que ejecutar para alcanzar un objetivo.

La representación abstracta de la interfaz puede considerase como un sistema de negociación en el diseño de la interfaz de usuario, de forma que:

- El diseñador gráfico puede realizar un trabajo creativo a nivel visual de la interfaz de usuario, sin depender de detalles de implementación.
- El analista puede, junto con el diseñador gráfico, comprobar la correctitud y completitud de la interfaz de usuario en relación con las tareas identificadas durante el análisis.
- El usuario puede opinar acerca de la adecuación del diseño de la interfaz de usuario con respecto al uso que de la misma va a realizar.

Además, esta representación permite la creación de herramientas que realicen su transformación desde la misma a una plataforma concreta de implementación de la interfaz con las ventajas que ello conlleva. Entre ellas puede destacarse el disponer de un sistema de prototipado inmediato dirigido a la apariencia real final no condicionada por widgets comerciales.

Capítulo 5

Análisis de la interfaz de usuario

En el capítulo anterior se ha planteado un sistema de representación abstracto de la interfaz de usuario. Con este sistema de representación se puede realizar una descripción de la apariencia de los componentes individuales de la interfaz. Así, cualquier diseñador de aplicaciones puede disponer de una definición única de la apariencia de los componentes que tiene que utilizar en su plataforma de desarrollo. Esto es posible gracias a la utilización de primitivas gráficas estándar.

Además, se ha incorporado en la representación un mecanismo de descripción de la composición visual de los componentes de la interfaz entre sí.

Se ha definido también un método para la descripción del diálogo que se produce entre componentes de la interfaz, de forma que se pueden representar las acciones del usuario en la interfaz.

En el presente capítulo se aborda la identificación de aspectos de la interfaz de usuario que, a partir de la representación abstracta propuesta, puedan cuantificarse automáticamente. Esta cuantificación tiene especial importancia puesto que puede realizarse desde la representación abstracta sin que sea necesario construir la interfaz para ello.

La elección de los análisis contemplados se ha realizado intentando abarcar aspectos de percepción por parte del usuario, de la apariencia visual de la interfaz de usuario (relacionados con estudios acerca de la aceptación de las interfaces por parte de los usuarios) y cognitivos del usuario.

La realización de los análisis de la interfaz de usuario, además de realizarse antes

de la implementación, tiene otra serie de ventajas como es que los resultados son cuantitativos y por lo tanto libres de subjetividad (lo que los hace muy útiles para la comparación entre interfaces de usuario). Por otro lado, puede estudiarse la relevancia que tiene en la interacción la utilización de componentes independientes sobre los que el usuario puede realizar operaciones visuales, debido a que pueden solaparse unos con otros. Otra ventaja es la identificación directa de situaciones anómalas sobre los estados de la interfaz, basándose en las transiciones entre estados. Por último, la determinación de tareas posibles, entendidas como combinación de acciones útiles del usuario sobre la interfaz, pueden proporcionar una herramienta muy útil para la validación de los requerimientos de la intefaz.

En este capítulo se indican los métodos de análisis de los siguientes conceptos:

- Estados y tareas de la interfaz.
- Jerarquía visual. Identificación de jerarquías posibles para cada estado y acciones inaccesibles dada una jerarquía particular.
- Diseño de la interfaz. Se han considerado dos elementos particulares a la hora de analizar el diseño: la tipografía y el color. Podrían analizarse otros muchos elementos: formas, espacios negativos, formatos de texto, etc. pero se han estudiado estos dos debido a que son elementos muy destacados e inmediatos en el diseño.
- Estética. Basados en la investigación de Ngo [NT00] sobre la teoría matemática de la estética se han construído los algoritmos para el estudio de cuatro de los parámetros de dicha investigación. Con ello se pretende demostrar la validez de la representación abstracta para estos estudios.
- Cardinalidad de componentes. La cardinalidad de los componentes tiene relación con aspectos cognitivos del usuario y nos permite determinar, según el límite de 7+-2 ítems de información manejable por la memoria a corto plazo del usuario, componentes contenedores con excesivos componentes contenido.
- Redimensionamiento. El análisis de la operación visual de cambio de tamaño es importante debido a que, para un determinado estado de interfaz, la utilización de dicha operación visual por parte del usuario puede llevarnos a perder parte de las acciones que éste puede realizar en el estado. Sirve de indicación para el diseñador sobre los límites de redimensionamiento qeu debe permitir al usuario.

Los algoritmos que se presentan para cada uno de los análisis a realizar sirven como indicación para la implementación de los mismos. Estos algoritmos utilizan las representaciones abstractas definidas anteriormente y generan nuevas representaciones que son utilizadas en otros análisis. Son como guías de implementación, no corresponden con algoritmos de detalle directamente implementables en un lenguaje de programación sino que determinan de forma general los pasos a seguir y sobre qué información utilizar para obtener la información cuantitativa que se busca.

5.1 Estados y tareas

5.1.1 Estados de la interfaz

A partir de la representación abstracta de esta propuesta de la interfaz pueden conocerse cuáles son las diferentes apariencias (combinación de apariencias de componentes individuales) de la interfaz con el fin de poder estudiarlas individualmente. Para ello, a partir de la representación abstracta de la composición de la interfaz, junto con la representación de la apariencia y el diálogo de sus componentes, pueden obtenerse, tal y como se indicó en el capítulo anterior, los estados que componen la interfaz y los comandos o acciones de usuario que provocan la aparición de estos estados.

El algoritmo de obtención de los estados y sus transiciones es el siguiente:

Procedimiento Estados de la interfaz

ENTRADA
- D1 : Representación de los componentes de la interfaz
- D2 : Representación de la composición de la interfaz
- D3 : Representación del diálogo de los componentes de la interfaz

SALIDA
- D4 : Representación de los estados de la interfaz
- D5 : Representación de las transiciones entre estados de la interfaz

Identificar el estado inicial
Estado estudio = Estado inicial
Almacenar estado estudio D4

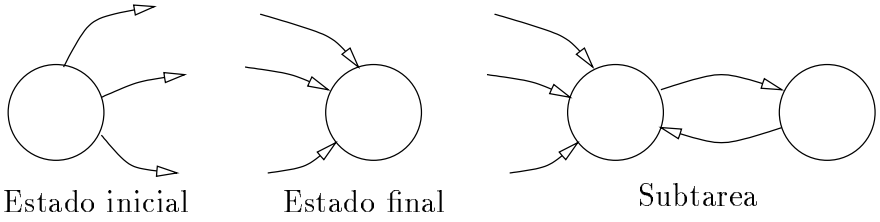

Figura 5.1: Transiciones relevantes entre estados

Hasta No más estados D4 **hacer**
Identificar los componentes activos con diálogo
Hasta No más componentes activos**hacer**
Hasta No más diálogo en estado estudio **hacer**
Si Componente activo tiene diálogo D3
Entonces
Determinar estado respuesta
Establecer y almacenar D4 nuevo estado o estado existente
Establecer transición D5 a nuevo estado o estado existente
FinSi
Siguiente diálogo
FinHasta
Analizar siguiente componente activo en estado actual
FinHasta
Estado estudio = siguiente estado D4
FinHasta
Fin Procedimiento

5.1.2 Estudio particular de las transiciones entre estados

Una vez que se han obtenido los estados de la interfaz y las transiciones entre estados puede hacerse un estudio preliminar acerca de la consistencia de la interfaz de usuario.

Una interfaz de usuario debe tener un único estado inicial y un único estado final. El estado final permite al usuario cerrar la aplicación con la cual está trabajando. Para identificar un estado inicial o final deben estudiarse los arcos de entrada y salida para cada estado (Figura 5.1).

El estado final será aquel en el cual todos sus arcos son de entrada. Si al estudiar los estados y sus transiciones se encuentra más de un estado final, debe concluirse que la interfaz no está bien diseñada.

Dada la propia naturaleza de la representación abstracta de la interfaz aquí presentada parece imposible que exista más de un estado inicial de la interfaz. De todas formas, puede comprobarse que sólo exista un estado que posea arcos de salida y los de entrada sean siempre desde estados que no se pueden alcanzar sin pasar por él mismo, con lo cual se garantiza un único estado inicial en la interfaz.

Un caso de transición entre estados que tiene una gran importancia es el correspondiente a un estado que posee únicamente un arco de entrada desde otro estado y un arco de salida hacia ese otro estado. Este tipo de transición identifica una subtarea que se lleva a cabo por parte del usuario, y se corresponde con un componente modal en las interfaces de manipulación directa.

Otra posible interpretación de esta última transición identificada es la posibilidad de realización de un *UNDO* sobre la interfaz. La representación no contiene ninguna semántica acerca de la intencionalidad de las acciones del usuario; por ello no se puede realizar interpretación alguna acerca de la significación de un arco. Lo que sí puede realizarse es la identificación de arcos contrarios. Es decir, para cada arco de transición de un estado a otro se puede comprobar la existencia de un arco contrario que permita una vuelta atrás desde el estado destino. Una posible utilidad de este estudio puede verse mejor en la sección siguiente, aplicándolo sobre las tareas ya que el *UNDO* se aplica sobre la realización de una acción del usuario.

5.1.3 Tareas posibles

En este análisis se obtienen las tareas posibles en la interfaz de usuario a partir de la representación abstracta.

Una vez obtenidos los estados posibles de la interfaz de usuario, junto con los eventos y componentes que provocan la transición, puede realizarse un análisis de las transiciones válidas entre el estado inicial de la interfaz y su estado final. Este conjunto de transiciones nos permitirá identificar tareas completas que no poseen secuencias de comandos repetidas. Cada una de estas tareas o comandos corresponden con acciones que el usuario puede realizar en la interfaz. Los comandos repetidos dentro de una

tarea posible corresponden con acciones del usuario que no incorporan nuevas funcionalidades a la tarea en cuestión. La determinación de las tareas posibles se realizará mediante un algoritmo, utilizando la representación de las transiciones entre estados (D5). El algoritmo es el siguiente:

Procedimiento Tareas posibles

ENTRADA

D4 : Representación de los estados de la interfaz

D5 : Representación de las transiciones entre estados

SALIDA

D6 : Representación de las tareas mínimas de la interfaz

Identificar el estado inicial

Marcar Estado Inicial como bifurcable

Estado estudio = Estado Inicial

Hasta (No más estados bifurcables) **hacer**

Hasta (Fin transiciones estado estudio) **hacer**

Cálculo siguiente estado

Si (Siguiente estado repetido en tarea) y (conjunto de estados intermedio repetidos en tarea)

Entonces Eliminar hasta anterior ocurrencia del estado repetido

Sino

Almacenar comando

Marcar Siguiente estado como bifurcable

Estado estudio = siguiente estado

FinSi

FinHasta

Fin Tarea

Estado estudio = ultimo estado bifurcable

Almacenar Nueva tarea D6

FinHasta

Fin Procedimiento

El resultado final de este algoritmo nos permite la representación de tareas D6 (secuencias de comandos posibles que podemos obtener a partir de la representación abstracta de la interfaz de usuario). Un ejemplo de las tareas identificadas mediante

este algoritmo es, para la interfaz utilizada en la descripción de diálogo de componentes (figura 4.7), el siguiente:

0(salir#LeftClick)3
0(B#LeftClick)1(salir#LeftClick)3
0(A#LeftClick)2(salir#LeftClick)3
0(B#LeftClick)1(F#LeftClick)5(salir#LeftClick)3
0(B#LeftClick)1(A#LeftClick)4(salir#LeftClick)3
0(A#LeftClick)2(B#LeftClick)4(salir#LeftClick)3
0(A#LeftClick)2(F#LeftClick)5(salir#LeftClick)3
0(B#LeftClick)1(A#LeftClick)4(F#LeftClick)5(salir#LeftClick)3
0(B#LeftClick)1(F#LeftClick)5(A#LeftClick)2(B#LeftClick)4(salir#LeftClick)3
0(A#LeftClick)2(F#LeftClick)5(B#LeftClick)1(A#LeftClick)4(salir#LeftClick)3

Una vez que las tareas posibles de la interfaz están identificadas puede hacerse un identificación de subtareas utilizando el concepto de *UNDO*, tal y como se ha descrito en la sección anterior. Para cada una de las tareas posibles se comprueba si existe vuelta atrás para cada uno de los comandos que forman parte de la tarea.

Como un comando es la ejecución de un evento sobre un componente de un estado, para cada uno de los comandos que provoca una transición entre estados se puede comprobar si existe un arco inverso. Con la existencia de este arco puede existir la posibilidad de realizar un *UNDO* (deshacer) del comando aunque, como ya se indicó en el capítulo cuatro, no es factible hacer una interpretación de la significación del arco inverso.

Generalizando el concepto de arco inverso a un arco dado entre dos estados podría utilizarse para identificar aquella secuencia o subsecuencia de comandos que posee un arco inverso entre el estado inicial y final de la secuencia. A la secuencia de aquellos estados unidos con arcos en una dirección sin arcos de retorno lo denominaremos **tarea transacción**, dado que, o bien se ejecuta toda la secuencia o se vuelve al primer estado de la secuencia. Un ejemplo de esta situación puede verse en la figura 5.2.

Para el análisis de las subtareas (*subtareas transacción*) y los estados con y sin arco de regreso al estado anterior se utiliza el siguiente algoritmo:

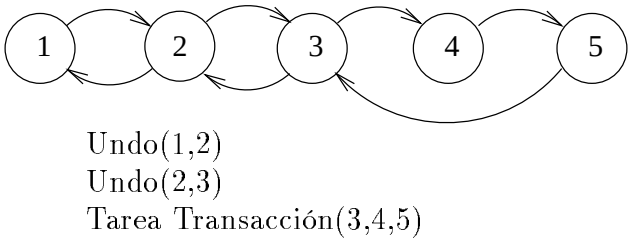

Figura 5.2: Ejemplo de arcos inversos y tarea transición

Procedimiento Operaciones Undo

ENTRADA

D5 : Representación de las transiciones entre estados de la interfaz

D6 : Representación de las tareas posibles de la interfaz

SALIDA

Operaciones Undo: Estados con vuelta al estado anterior y tareas transacción

Identificar primera tarea en D6

Hasta Fin Tareas en D6 **hacer**

Identificar primer Estado Tarea

Hasta Fin Estados en Tarea **hacer**

Si (Transición en D5 al estado anterior)

Entonces

Almacenar Estado Actual y Estado Anterior

Sino

Almacenar Falta transición Estado Actual y Estado Anterior

Buscar en D5 transición a algún estado anterior de la tarea

Almacenar transición Estado Actual y Estado Tarea más cercano al actual

FinSi

Siguiente Estado Tarea

FinHasta

Siguiente Tarea

FinHasta

Fin Procedimiento

5.1.4 Componentes sin diálogo

Existen componentes que no tienen interacción directa con las acciones de usuario. Es el caso de aquellos cuya utilidad básica es mostrar información, como mensajes de texto o iconografía.

Los componentes contenedores también son, en la mayoría de los casos, componentes sin acciones de usuario asociadas a nivel de interfaz, aunque pueden tener asociada la posibilidad de realización de acciones de usuario sobre las operaciones visuales. Dado que las operaciones visuales no son dependientes de la interfaz que se está representando, no son relevantes a la hora de su análisis. Su única intervención se detallará en el análisis de la jerarquía visual de la interfaz.

A partir de la representación abstracta definida es posible obtener los componentes de la interfaz que no tienen asociadas acciones de usuario, basándonos en el siguiente algoritmo:

Procedimiento Componentes sin acción

ENTRADA
D1 : Representación de los componentes de la interfaz
D3 : Representación del diálogo de los componentes de la interfaz
SALIDA
Lista de componentes : Información acerca de los componentes de la interfaz sin acciones asociadas

Hasta No más componentes D1 **hacer**
Buscar componente en D2
Si No diálogo para componente
Entonces
Añadir componente a la lista
FinSi
Siguiente componente en D1
FinHasta
Fin Procedimiento

5.2 Jerarquía de capas visuales

Dado que la representación propuesta nos permite conocer en cada instante el estado de la interfaz, los componentes que lo forman y los eventos posibles sobre cada uno de esos componentes, es posible realizar un análisis de la estructura jerárquica visual de la interfaz. Dentro de este análisis podremos, para cada estado, determinar el número de componentes de la interfaz activos esperando evento y cuáles de ellos son visibles en mayor o menor medida por parte del usuario.

Para obtener esta información deben examinarse los estados que poseen componentes contenedores activos y visibles al mismo tiempo, extrayendo la acción de usuario que la provocó con el fin de conocer el orden de aparición visual de los componentes para el estado.

En un estado cualquiera de la interfaz se puede saber cuál es el número de componentes contenedores independientes visibles que forman parte del mismo. Si es 1 no existe jerarquía visual. Si dicho número es mayor que 1 deben estudiarse las transiciones entre estados para determinar el número de posibles jerarquías visuales distintas que se pueden dar. Para ello, se denotará E_{AC} (estado actual) al estado que se está estudiando. Se denotará E_{EAC_i} al conjunto de estados anteriores de E_{AC} que, por la aparición de un C_i (componente contenedor independiente), provocan el estado E_{AC}

Si $E_{EAC_i} = 1$ entonces el componente contenedor C_i está en la capa de jerarquía visual superior. Si $E_{EAC_i} > 1$ entonces existen n componentes contenedores individuales que provocan la aparición del estado E_{AC}, lo cual da lugar a un máximo de 2^n posibles jerarquías visuales para el estado actual.

El número de posibles jerarquías visuales se puede reducir descartando aquellas combinaciones en las cuales, en la última transición al estado actual, el componente contenedor independiente que está en el nivel superior posee la propiedad de posición relativa. En este caso, el usuario podrá desplazarlo por el dispositivo de visualización para acceder a los componentes que se encuentran en niveles jerárquicos visuales más bajos.

Tras eliminar dichas jerarquías, quedan por examinar un número j de jerarquías visuales, donde $1 \geq j \geq 2^n$. Para cada una de ellas, se examinarán las áreas visuales ocupadas por cada uno de los componentes que forman parte del estado para determinar cuál es su área perceptible por el usuario con la cual podría interactuar.

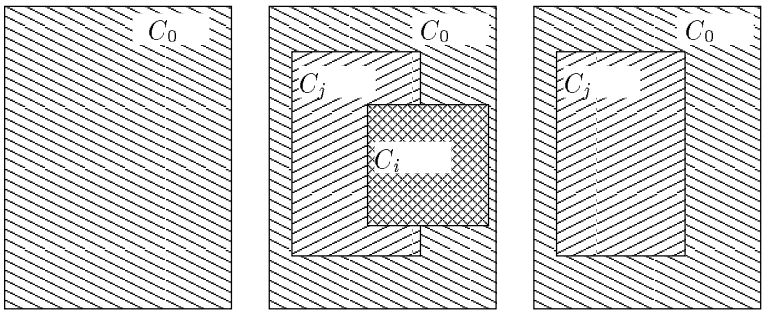

Figura 5.3: Posibles jerarquías visuales para E_{AC}

Para la determinación de las áreas visuales de cada componente debe establecerse el orden de aparición del componente en la jerarquía, y para ello se utilizarán la definiciones del diálogo de componentes y de la transición entre estados. Dado un estado E_{AC}, se cumple que para cada jerarquía existe un estado anterior E_{AN} donde una acción del usuario provoca la aparición en la interfaz de un componente contenedor independiente C_i. Teniendo en cuenta que para cada estado se conocen los componentes contenedores independientes que forman parte del mismo, puede construirse el conjunto de las transiciones entre estados y los componentes contenedores que los modifican de la siguiente forma:

$$E_{ANC_j} \rightarrow_{C_j} \rightarrow E_{ANC_i} \rightarrow_{C_i} \rightarrow E_{AC}$$

donde C_ϕ (componente contenedor del estado E_{ANC_j}), C_j y C_i son componentes que forman parte del estado E_{AC} y la secuencia de estados desde el estado E_{ANC_j} (Estado inicial) determina la jerarquía visual de E_{AC}. El componente $C_\phi \in E_{ANC_J}$. En la transición de estados mostrada no se indican los componentes y eventos que provocan la transición sino cuál es el componente contenedor independiente que entra a formar parte del estado y que se encuentra en el nivel jerárquico visual más alto. Pueden verse algunas de las posibilidades de jerarquía visual para el estado E_{AC} en la figura 5.3 para el caso de 3 componentes contenedores.

Una vez determinada la jerarquía visual puede hacerse un estudio de la misma con el fin de determinar si el usuario puede acceder a toda la funcionalidad de la interfaz.

Este estudio se realizará a través de la determinación de las áreas visibles y no visibles de cada componente contenedor independiente. Una vez que se determinen las áreas no visibles de los componentes contenedores podrán extraerse los componentes contenido de cada uno de ellos con propiedad Activo(T) y evento asociado que no son alcanzables por el usuario.

La determinación del área visible de cada componente contenedor está relacionada con el nivel jerárquico visual que ocupa. De esta forma, para el ejemplo de transición entre estados anterior **??**, en el cual existen tres componentes contenedores con un orden jerárquico, el área visual correspondiente a cada uno de ellos puede hallarse:

$Area_{Visible}(C_\phi) = Area(C_\phi) - ((Area(C_\phi) \bigcap Area(C_j)) + (Area(C_\phi) \bigcap Area(C_i)))$
$Area_{Visible}(C_j) = Area(C_j) - (Area(C_j) \bigcap Area(C_i))$
$Area_{Visible}(C_i) = Area(C_i)$

Este cálculo puede generalizarse para cualquier componente contenedor independiente que ocupe el nivel jerárquico m:

$$Area_{Visible}(C_m) = Area(C_m) - \sum_{i=0}^{m-1} (Area(C_m) \bigcap Area(C_i)) \tag{5.1}$$

Una vez conocido el área visible de cada componente contenedor podemos obtener el área no visible:

$$Area_{NoVisible}(C_m) = Area(C_m) - Area_{Visible}(C_m) \tag{5.2}$$

La determinación de aquellas acciones no accesibles por el usuario en la interfaz para una jerarquía concreta de un estado E_{AC} corresponde con la identificación de los componentes contenido activos que pertenecen al área no visible de su componente contenedor.

$$C_i \in Area_{NoVisible}(C_m)/C_i \text{componente Activo(T) de } E_{AC}$$

El algoritmo es el siguiente:

Procedimiento Jerarquías visuales

ENTRADA

D3 : Representación del diálogo de componentes

D4 : Representación de los estados de la interfaz

D5 : Representación de las transiciones entre estados

SALIDA

Jerarquía visual : Información por estado sobre la no alcanzabilidad de componentes activos debido a la jerarquía visual.

Hasta Fin Estados D4 **hacer**

NumCont = componentes contenedores independientes del estado

Si (NumCont > 1)

Entonces

Hallar Caminos hasta Estado con Comp. Contenedores Ind.

Hasta Fin Caminos **hacer**

Cálculo $Area_{NoVisible}$ Comp. Contenedores

Si $(Comp.ContenidosActivo(T) \in Area_{NoVisible})$

Entonces

Mostrar Comp. Contenedor y contenidos inaccesibles

FinSi

FinHasta

FinSi

Siguiente Estado

FinHasta

Fin Procedimiento

5.3 Diseño de la interfaz

5.3.1 Tipografía

En esta sección se examina la tipografía existente en la interfaz con el fin de determinar su consistencia a nivel de fuentes utilizadas, su tamaño, su color e incluso su mensaje. Para ello, puede hacerse una identificación por estado, con el fin de comprobar textos repetidos en un estado, o bien a nivel de definición de componentes para comprobar que un mismo texto tenga asociado la misma acción en todos los estados de la interfaz.

Procedimiento Tipografía

ENTRADA

D1 : Representación componentes de la interfaz

D2 : Representación de la composición de la interfaz

D3 : Representación del diálogo de componentes

D4 : Representación de los estados de la interfaz

SALIDA

Estudio de tipografía: Determinación de parámetros tipográficos

Hasta Fin componentes D1 **hacer**

Si (Componente de texto)

Entonces

Obtener Nombre componente, Texto, Fuente, TamañoFuente, ColorFuente

Marcar valores distintos de atributos

FinSi

FinHasta

Mostrar todos los valores de componentes texto distintos

Hasta Fin de estados D4 **hacer**

Hasta Fin componentes estado **hacer**

Si (componente texto en componente activo)

Entonces

Marcar componente texto, evento, respuestas

FinSi

Mostrar valores componentes texto

FinHasta

Siguiente estado

FinHasta

Fin Procedimiento

5.3.2 Color

En esta sección se realiza un estudio de la composición cromática de los estados de la interfaz de usuario. Para ello, se realiza una recopilación de los colores utilizados como relleno de componentes y de las fuentes utilizadas. La recopilación de colores

de fuentes en este estudio se utiliza para determinar su adecuación al color de los componentes en donde están incluidas, y su adecuación al resto de colores de la interfaz. Se realiza una recopilación de la participación de colores para cada componente, indicando su grado de proporcionalidad en el componente que lo engloba. Una interpretación del significado de los colores dentro de la interfaz es una tarea imposible debido a que, desde un punto de vista artístico, la significación de los colores es muy subjetiva y sujeta a variables tan distintas que no se puede preveer la intencionalidad del diseñador en el momento de crear la interfaz.

Procedimiento Color

ENTRADA

D1 : Representación componentes de la interfaz

D2 : Representación de la composición de la interfaz

D4 : Representación de los estados de la interfaz

SALIDA

Estudio de color: Determinación de participación cromática en la interfaz

Hasta Fin Estados D4 **hacer**

Hasta (Fin componentes estado) **hacer**

Si (componente texto)

Entonces

Marcar ColorFuente, ColorRelleno componente contenedor

Sino

Calcular Area componente

Marcar Area componente, ColorRelleno componente

FinSi

FinHasta

Mostrar valores marcados

Cálculo proporción (Areas, ColorRelleno)(Area total estado)

Mostrar proporciones para cada color

Siguiente estado

FinHasta

Fin Procedimiento

5.4 Propiedades Visuales

Es interesante la realización de una medida cuantitativa de parámetros estéticos porque permite una comparación objetiva entre distintas interfaces. Basándose en los trabajos de Ngo [NT00], se han elaborado los procedimientos necesarios para, a partir de la representación abstracta de la interfaz, realizar un cálculo cuantitativo de algunas de las medidas estéticas desarrolladas por éste.

5.4.1 Medida de simplicidad

La simplicidad se basa en la sencillez de la forma, es una combinación de elementos que da lugar a una comprensión sencilla del significado de un patrón. La simplicidad en el diseño de pantallas se alcanza optimizando el número de elementos en pantalla y los puntos de alineamiento. Tullis [Tul84] presenta una medida de la complejidad de pantallas en modo texto, basada en el trabajo de Bonsiepe [Bon68], quien propone un método para medir la complejidad de las páginas diseñadas tipográficamente mediante la aplicación de la teoría de la información. Esto incluye contar el número de filas o columnas diferentes en la pantalla que se usan como posiciones iniciales de elementos alfanuméricos. La teoría de la información se utiliza para calcular la complejidad de la disposición de las posiciones iniciales.

Un método de cálculo más simple es:

$$SMM = \frac{3}{n_{vap} + n_{hap} + n} \in [0, 1] \tag{5.3}$$

donde

n_{vap} y n_{vap} son los valores de los puntos de alineación horizontal y vertical, y n es el número de objetos en el objeto contenedor.

El algoritmo para el cálculo de simplicidad sobre un estado es el siguiente:

Procedimiento Simplicidad

ENTRADA

D4 : Estado de la interfaz

SALIDA

Medida de simplicidad : Medida de la simplicidad de los estados

Hasta Fin de estados D4 **hacer**
Inicializar Número componentes, Número Posiciones X e Y
Hasta Fin de componentes estado **hacer**
Incrementar Número de componentes
Si componente visible
Si Posición Inicial X Nueva
Entonces Incrementar Número Posiciones X diferentes
FinSi
Si Posición Inicial Y Nueva
Entonces Incrementar Número Posiciones Y diferentes
FinSi
FinSi
FinHasta
Cálculo y muestra de datos de Simplicidad del estado
FinHasta
Fin Procedimiento

5.4.2 Medida de densidad

La densidad mide el grado en que la pantalla está cubierta con objetos. Se calcula minimizando los niveles de densidad de pantalla. Una medida de densidad, indicada por Tullis, es el porcentaje de posiciones de texto existentes en el objeto contenedor que contienen datos.

En lugar de buscar caracteres, nuestra medida incluye componentes de la siguiente forma:

$$DM = 1 - \frac{\sum_i^n a_i}{a_{frame}} \in [0, 1] \tag{5.4}$$

donde

a_i y a_{frame} son las áreas del componente i y del componente contenedor, y n es el número de componentes en el componente contenedor.

El algoritmo de cálculo de densidad es el siguiente:

Procedimiento Densidad

ENTRADA

D4 : Estados de la interfaz

SALIDA

Medida de densidad : Medida de la densidad de los componentes contenedores del estado

Hasta Fin de estados D4 **hacer**

Hasta Fin de componentes contenedores visibles estado **hacer**

Inicializar Número componentes, Área contenidos y Área contenedor

Cálculo Área contenedor

Hasta Fin de componentes contenido del estado **hacer**

Incrementar Número de componentes

Área contenidos = Área contenidos + Cálculo área componente

FinHasta

Cálculo y muestra de medida densidad de componente contenedor

FinHasta

FinHasta

Fin Procedimiento

5.4.3 Medida de economía

La medida de economía (Figura 5.4) se relaciona con el uso cuidadoso y discreto de elementos visuales para transmitir el mensaje del modo más simple posible. La economía se alcanza utilizando el menor número posible de estilos, técnicas visuales y colores.

La economía por definición corresponde con:

$$ECM = \frac{3}{n_{tam} + n_{color} + n_{forma}} \in [0, 1] \tag{5.5}$$

donde

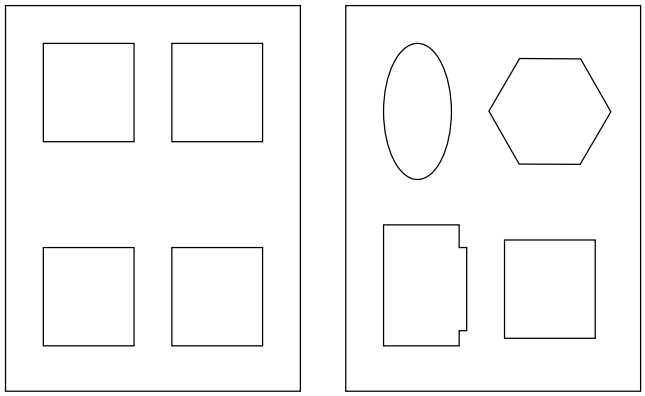

Figura 5.4: Buena y mala interfaz según medida de economía

n_{tam} es el número de tamaños distintos de componentes del estado, n_{color} es el número de colores diferentes en los componentes y n_{forma} es el número de formas o figuras diferentes del estado.

El algoritmo para el cálculo de la economía es el siguiente:

Procedimiento Economía

ENTRADA

D4 : Estados de la interfaz

SALIDA

Medida de Economía : Medida de la economía de los componentes contenedores de cada estado

Hasta Fin de estados D4 **hacer**

Hasta Fin de componentes contenedores visibles estado **hacer**

Inicializar Número componentes, Número formas, colores y tamaños

Hasta Fin de componentes contenido componente contenedor **hacer**

Actualizar Número componentes

Actualizar Número formas, colores y tamaños

FinHasta

Cálculo y muestra de medida economía de componente contenedor

FinHasta

FinHasta

Fin Procedimiento

5.4.4 Medida de Regularidad

La medida de regularidad se relaciona con la uniformidad de los elementos en base a algún principio. La regularidad en el diseño de pantallas se alcanza estableciendo un espaciado consistente y estándar entre los puntos de alineación horizontales y verticales de los elementos de pantalla, y minimizando dichos puntos.

La regularidad, por definición, es una medida de la uniformidad de una pantalla y viene dada por:

$$RM = \frac{|RM_{alineamiento}| + |RM_{espaciado}|}{2} \in [0,1] \tag{5.6}$$

$RM_{alineamiento}$ es el grado de minimización de los puntos de alineamiento siendo

$$RM_{alineamiento} = 1 - \frac{n_{vap}+n_{hap}}{2n}$$

y $RM_{espaciado}$ es el grado de equiespaciado entre los puntos de alineación siendo

$$RM_{espaciado} = \begin{cases} 1 & \text{si } n = 1, \\ 1 - \frac{n_{espaciado}-1}{2(n-1)} & \text{en otro caso} \end{cases}$$

donde

n_{vap} y n_{hap} son los valores de los puntos de alineación y vertical,

$n_{espaciado}$ es el número de distancias diferentes entre los puntos iniciales de fila y columna, y

n es el número de componentes en el componente contenedor.

Procedimiento Regularidad

ENTRADA

D4 : Estados de la interfaz

SALIDA

Medida de Regularidad : Medida de la regularidad de los componentes contenedores de cada estado

Hasta Fin de estados D4 **hacer**

Hasta Fin de componentes contenedores visibles estado **hacer**

Inicializar Número componentes, n_{vap}, n_{hap}, $n_{espaciado}$

Hasta Fin de componentes contenido componente contenedor **hacer**

Actualizar Número componentes

Actualizar n_{vap}, n_{hap}, $n_{espaciado}$

FinHasta

Cálculo $RM_{espaciado}$

Cálculo y muestra de medida de regularidad del componente contenedor

FinHasta

FinHasta

Fin Procedimiento

5.5 Cardinalidad de componentes

En esta sección se examina el número de componentes contenido visibles que existe en cada componente contenedor. Se realiza además un conteo general de los componentes (tanto contenedores como contenido) que existen para cada estado.

El algoritmo de obtención de la cardinalidad de los componentes de los estados de la interfaz es el siguiente:

Procedimiento Cardinalidad

ENTRADA

D4 : Estados de la interfaz

SALIDA

Medida de Cardinalidad : Medida de la cardinalidad de los estados

Hasta Fin de estados D4 **hacer**

```
        Inicializar Número componentes
        Hasta Fin de componentes contenedores visibles estado hacer
            Calcular Número componentes contenido
            Mostrar cardinalidad de componente contenedor
        FinHasta
        Mostrar cardinalidad del estado
    FinHasta
Fin Procedimiento
```

5.6 Redimensionado

A la hora de realizar operaciones visuales con los componentes contenedores de la interfaz debe tenerse en cuenta que se trabaja sobre un dispositivo finito y discreto. El valor de la propiedad de posición que un componente contenedor puede poseer está limitado por el tamaño y resolución del dispositivo de visualización en el cual se muestra.

A la hora de estudiar la operación visual de redimensionamiento de componentes deben establecerse cuáles son los límites superior e inferior de cada componente. El límite superior es fácil de establecer y corresponde con el tamaño en píxeles del dispositivo de visualización. En cuanto al límite inferior, deben tenerse en cuenta las propiedades de tamaño de los componentes contenido del componente contenedor.

Si todos los componentes contenido del componente contenedor en estudio tienen la propiedad de tamaño relativo (caso muy poco común), el límite inferior viene dado por la capacidad de percepción del usuario. Cuanta mayor capacidad de percepción a nivel visual tenga el usuario menor tamaño podrá dar al componente contenedor y seguir interactúando con él.

En el caso más común, en el cual existen componentes contenido que tienen la propiedad de tamaño fijo, el tamaño mínimo del componente contenedor viene dado por los tamaños fijos de los componentes contenidos. Para cada eje del componente contenedor debe obtenerse la suma de los tamaños de los componentes contenido que coinciden en algún punto del eje perpendicular. Por ejemplo, para calcular el tamaño mínimo del eje X, deben sumarse los tamaños de los componentes contenido con algún punto en común en el eje Y (Figura 5.5). Esto puede expresarse matemáticamente de

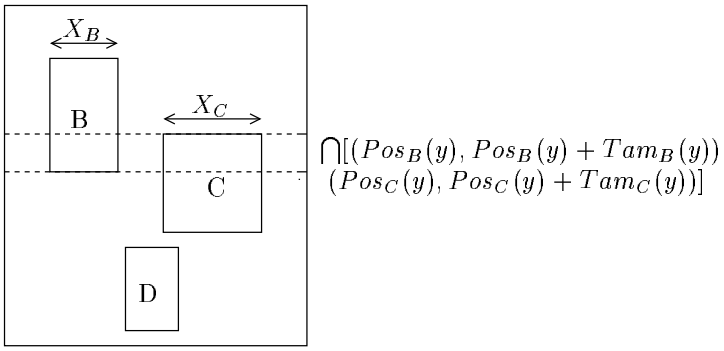

Figura 5.5: Ejemplo sumatorio tamaño componentes contenido

la siguiente forma:

Para un componente contenedor C y sus p componentes contenido C_i, el tamaño mínimo en el eje X es $(Tam_{min}(X))$,

$$Tam_{minC}(X) = Max \sum_{i=1}^{p} Tam_{C_i}(x) \tag{5.7}$$

$$\forall C_i \rightarrow \bigcap [Pos_{C_i}(y), (Pos_{C_i}(y) + Tam_{C_i}(y))] \neq \phi$$

Lo mismo se aplica al tamaño mínimo en el eje Y $(Tam_{min}(Y))$.

$$Tam_{minC}(Y) = Max \sum_{i=1}^{p} Tam_{C_i}(y) \tag{5.8}$$

$$/\forall C_i \rightarrow \bigcap [Pos_{C_{i(x)}}, (Pos_{C_i}(x) + Tam_{C_i}(x))] \neq \phi$$

Esta fórmula de cálculo funciona bien en los casos en donde los componentes contenido tiene tamaño fijo y su propiedad de posición es relativa en Centrado y Equiespaciado, puesto que no existen restricciones de valor exacto con respecto a su posición en el componente contenedor. Sin embargo, en los casos en donde la posición es relativa y

se aplica Justificado() o Alineado() con un valor concreto en píxeles, no es correcta. Para ello debe añadirse en la fórmula la restricción de posición que debe mantener el componente contenido.

$$Tam_{minC}(X) = Max\left(\sum_{i=1}^{p}(Tam_{C_i}(x) + Justificado_{C_i}(x) + Alineado_{C_i}(x))\right) \tag{5.9}$$

Por último queda un caso por contemplar que implica un estudio para cada uno de los posibles tamaños del componente contenedor. En el caso de que el componente contenido posea la propiedad de posición relativa en un valor (x,y) implica que dependiendo del tamaño del componente contenedor tendrá una posición u otra. Si se une esto a la característica de tamaño fijo, tendremos un tamaño de componente contenedor mínimo en el cual el componente contenido permanece dentro del componente contenedor, y por debajo del cual parte del componente contenido sale fuera de los límites del componente contenedor.

La propiedad de posición relativa responde a la siguiente formulación: Para un componente contenedor A, la posición relativa de un componente contenido B responde a una constante: $\frac{Pos_B(x)}{Tam_A(x)}$. Si el tamaño del componente contenedor A cambia a A', entonces $Pos_{B'}(x) = Tam_{A'}(x)\frac{Pos_B(x)}{Tam_A(x)}$. Como se trata de un valor constante se denominará $ConstPos_{C_i}(x)$ a la constante de posición relativa del componente C_i en el eje X y $ConstPos_{C_i}(y)$ a la constante de posición relativa del componente C_i en el eje Y.

Para el cálculo del tamaño mínimo del componente contenedor en el caso de poseer componentes contenido de tamaño fijo y posición relativa tenemos entonces:

$$Tam_{minC}(X) = Max\left(\forall posiblesTam_C\left(\sum_{i=1}^{p}(Tam_{C_i}(x) + (Tam_C(x)ConstPos_{C_i})\right)\right) \tag{5.10}$$

Es posible simplificar la ecuación anterior. Atendiendo a que el tamaño en el eje X de un componente contenido es

$$(Tam_{C_i}(x) + (Tam_C(x)ConstPos_{C_i}(x))$$

si solo existiese un componente contenido C_i se podrían eliminar el sumatorio y el máximo, con lo cual:

$$Tam_{minC}(x) = (Tam_{C_i}(x) + (Tam_C(x)ConstPos_{C_i}(x))$$

donde $Tam_{minC}(x)$ y $Tam_C(x)$ son el mismo valor.

Se despeja el valor de $Tam_C(x)$:

$$Tam_{minC}(x) - Tam_C(x)ConstPos_{c_i}(x) = Tam_{C_i}(x)$$
$$Tam_C(x)(1 - ConstPos_{C_i}(x)) = Tam_{C_i}(x)$$
$$Tam_C(x) = \frac{Tam_{C_i}(x)}{(1-ConstPos_{C_i}(x))}$$

Se sustituye en la formulación original:

$$Tam_{minC}(X) = Max\left(\sum_{i=1}^{p}(Tam_{C_i}(x) + ((\frac{Tam_{C_i}(x)}{(1 - ConstPos_{C_i}(x))})ConstPos_{C_i}(x))\right) \tag{5.11}$$

El algoritmo para el cálculo de los tamaños mínimos de los componentes contenedores en un estado, que satisfagan las propiedades de tamaño y posición de los componentes contenido es el siguiente:

Procedimiento Tamaños Mínimos Contenedores

ENTRADA

D4 : Estados de la interfaz

SALIDA

Medida redimensionamiento : Tamaños mínimos de los componentes contenedores

Hasta Fin de estados D4 **hacer**

Identificar componentes contenedores visibles con propiedad Tamaño Relativo en estado

Hasta Fin contenedores identificados estado **hacer**

Identificar componentes contenido tamaño fijo

Hasta Componente contenedor principal**hacer**

Calcular tamaño mínimo componente contenedor

Repercutir tamaño en componente contenedor superior si hubiera

FinHasta

Mostrar tamaño mínimo componente contenedor

FinHasta

FinHasta

Fin Procedimiento

5.7 Conclusiones del capítulo

En este capítulo se han planteado una serie de análisis representativos sobre aspectos de la interfaz de usuario como son percepción, apariencia visual y procesos cognitivos. Se han elegido estos aspectos por ser los más descriptivos, pero que pueden ser incrementados con otros análisis que investigadores de distintos campos puedan realizar sobre la representación. De hecho, es un método adecuado para el estudio de la incidencia de los conceptos estudiados en la aceptación o no de una interfaz de usuario por parte de los usuarios combinándolo con un sistema de encuestas.

Las contribuciones de este capítulo son:

1. Sobre los estados y tareas de la interfaz. Se evalúa la interfaz a nivel de estados que puede alcanzar y las acciones que permiten realizar una transición de un estado a otro. La combinación de las acciones conduce a una determinación de tareas posibles en la interfaz.

    - Determinación de los estados de la interfaz de usuario. Cada estado es único en función de los componentes que posee y las propiedades de sus componentes.
    - Determinación de las transiciones entre estados. Puede establecerse un grafo de transiciones entre estados que identifica las acciones de usuario que provocan las transiciones y componentes afectados.
    - Determinación de estados iniciales y finales. Permite la identificación de situaciones anómalas como pueden ser varios estados iniciales y/o finales.
    - Determinación de estados modales. Transición de estados que sólo permite la salida del estado una vez que se ha entrado en él. La determinación de este tipo de estados permite identificar subtareas.
    - Determinación de tareas posibles en la interfaz. Interpretando las transiciones entre estados se determinan las tareas posibles de la interfaz, entendiendo como tarea la conjunción de acciones dentro de la interfaz que conducen al usuario desde un estado inicial a un estado final.

2. Sobre la percepción del usuario.

    - Determinación de la participación de elementos cromáticos en la interfaz.

Se realiza un estudio acerca de los colores que se pueden hallar en la interfaz y cual es su proporción para cada estado.

- Aparición de la tipografía en la interfaz, determinando todos los mensajes de texto que aparecen en la interfaz, los tipos de fuente, colores de fuente y tamaño.
- Análisis de consistencia de acciones sobre textos iguales.
- Determinación de tamaños mínimos de los componentes de la interfaz de manera que conserven las propiedades definidas para los mismos y la posibilidad de alcanzar las acciones que ofrecen.

3. Sobre la estética de la interfaz. Determinación de una medida cuantitativa acerca de cuatro parámetros estéticos dentro de la interfaz (simplicidad, densidad, economia y regularidad).

4. Sobre los procesos cognitivos.

   - Determinación de la cardinalidad de la interfaz. Se realiza mediante la medida de la cardinalidad de los componentes contenedores de la interfaz y a partir de ella se obtiene la cardinalidad de estados.
   - Determinación de acciones inaccesibles. Estas acciones inaccesibles pueden provocarse tanto por errores en el diseño de la interfaz como por la incidencia de la jerarquía de capas visuales, al manejar la interfaz distintos componentes contenedores independientes en el mismo estado.

5. Sobre la jerarquía visual. Determinación para cada estado, si posee varios componentes contenedores independientes, de la colocación jerárquica de dichos componentes. Este es un aspecto no estudiado anteriormente en ningún modelo. Identifica las acciones no alcanzables en la interfaz debido a la colocación de componentes sobre el dispositivo de visualización. Se distinguen dos tipos de acciones de usuario: las **operaciones visuales**, que son aquellas que el usuario realiza sobre el tamaño y posición de componentes en un dispositivo de visualización, y que por lo tanto deberían ser independientes a la interfaz, y las propias **acciones** del usuario sobre la interfaz.

   - Determinación de la existencia o no de jerarquía visual en los estados de la interfaz.

- Determinación de las diferentes estructuras jerárquicas de los componentes contenedores independientes pertenecientes a un estado.
- Determinación de las acciones inaccesibles en el estado debido al orden jerárquico de los componentes contenedores.

Este análisis se ha realizado basándose en métodos cuantitativos porque proporcionan un sistema de comparación entre interfaces de usuario inexistente hasta el momento. Prácticamente todos los sistemas de evaluación de interfaces de usuario se centran en sistemas cualitativos, donde la evaluación final se realiza a partir de opiniones de expertos. La posibilidad de obtener una cuantificación de un conjunto de conceptos de la interfaz de modo automático proporciona un sistema autónomo y no subjetivo.

Capítulo 6

Conclusiones y trabajo futuro

6.1 Conclusiones

En el presente trabajo se han presentado los siguientes resultados:

- Una descripción del sistema interactivo, incluyendo a todos los participantes en el mismo, mediante la analogía del Modelo Humano-Procesador. Se han estudiado las formas de abordar el diseño de las interfaces. Se han clasificado los métodos de evaluación de interfaces de usuario atendiendo a si son cuantificables o cualificables. Por último, se ha determinado la importancia de los factores humanos desde el punto de vista del diseño visual, a partir de parámetros estéticos que influyen en la percepción de la interfaz por parte del usuario.(Capítulo 2)

- Una descripción de algunos de los modelos y herramientas actuales que permiten diseñar y especificar una interfaz de usuario, realizando una caracterización de los mismos y haciendo especial hincapié en aquellos modelos en los cuales podría establecerse una relación entre el modelo y la componente visual de la interfaz a nivel de presentación visual hacia el usuario. Se han detectado lagunas importantes en los modelos y herramientas para la representación de partes del diseño de la interfaz, en concreto el modelo de presentación abstracto y el modelo de composición abstracto. (Capítulo 3)

- Un sistema para representar la interfaz de usuario desde su componente visual, utilizando para ello una definición gráfica de los elementos que componen vi-

sualmente la interfaz. Se determina también un sistema de representación del diálogo entre los componentes individuales, en función de los eventos del sistema, y un sistema de representación del conjunto de componentes individuales de la interfaz que identifican un estado de la interfaz.

Se incluye un sistema de extracción de información que determina la existencia de una dependencia visual y jerárquica demostrable entre los mismos. Dentro de la dependencia jerárquica se identifican situaciones conflictivas a nivel de estado de interfaz, como la no alcanzabilidad de acciones posibles del usuario o la no visibilidad de componentes de la interfaz.

A partir de la identificación de estados de la interfaz y de sus transiciones se definen los diagramas de estados de la interfaz, a partir de los cuales se pueden estudiar las tareas que el usuario realiza sobre la interfaz.

Se establecen restricciones de posicionamiento a nivel de agrupación de componentes individuales y funcionamiento entre los componentes en respuesta a eventos. Se propone un sistema de almacenamiento de esta información de manera que sea exportable y por lo tanto la misma interfaz pueda construirse en diferentes sistemas.(Capítulo 4)

- Se han estudiado algunas propiedades que se pueden extraer de esta estructuración de la componente visual de la interfaz y se han indicado los mecanismos necesarios para su análisis y evaluación.(Capítulo 5). Se han mostrado los algoritmos desarrollados para la obtención de las propiedades inherentes a la interfaz desde su apariencia y comportamiento.

6.2 Trabajo Futuro

Dado que el presente trabajo se centra en un concepto poco desarrollado en los sistemas actuales de Diseño de Interfaces de Usuario, el trabajo que queda por desarrollar en muy amplio. Dentro de los objetivos más próximos que podemos identificar se encuentran:

- Desarrollo de uno o varios sistemas de transformación del modelo propuesto de representación para poder ser utilizado en metodologías de Diseño de Sistemas Interactivos, y así completar o verificar otras fases de diseño como el modelo mental de usuario, el modelo de tareas o el modelo de aplicación.

- Especificación de la representación de la interfaz mediante un marco teórico formal.
- Inclusión en la representación de pre-condiciones y post-condiciones para la ejecución de las acciones del usuario que forman el diálogo de componentes, dominios asociados a las funciones de entrada y salida de información en la interfaz, y la especificación de la información visual presentada al usuario desde la aplicación mediante metáforas, incluyendo restricciones topológicas y geométricas en la misma.
- Definición de plantillas de representación para la realización de guías de estilo y guías corporativas.
- Ampliación de los parámetros de estética en el análisis de la representación de interfaces de usuario.
- Inclusión en el sistema de representación de un método para relacionar las acciones de la interfaz que tengan repercusión con los objetos de la aplicación que no pertenezcan a la interfaz. Esto permitiría incorporar una técnica de definición abstracta de la interfaz a las técnicas utilizadas en Ingeniería de Software para el análisis y diseño de aplicaciones.
- Explorar la posibilidad de exportar la propuesta y ampliarla para permitir la representación de interfaces de usuario en tres dimensiones.

Bibliografía

[AAC+01] J. Abascal, I. Aedo, J. Cañas, M. Gea, J. Lorés, A. Martinez, M. Ortega, P. Valero, and M. Velez. *Introducción a la Interacción Persona-Ordenador*. 2001.

[Arc92] Arch. A metamodel for the runtime architecture of an interactive systems. *SIGCHI Bulletin*, 24(1):32–37, 1992.

[Asp91] M. Aspillaga. Screen design: A location of information and its effects on learning. *Journal of Computer-Based Instruction*, 18(3):89–92, 1991.

[BB97] R. Butterworth and A. Blandford. Programmable user models: The story so far. Puma Working paper 8. Available from http://www.cs.mdx.ac.uk, 1997.

[Bon68] G. Bonsiepe. A method of quantifing order in typographic design. *Journal of Typographic Research*, (2):203–220, 1968.

[BPLM98] R. Bastide, P. Palanque, D.-H. Le, and J. Muñoz. Integrating rendering specifications into a formalism for the design of interactive systems. In P. Markopoulos and P. Johnson, editors, *Design, Specification and Verification of Interactive Systems '98*, Eurographics, pages 171–190, Wien, 1998. Springer-Verlag. Proceedings of the Eurographics Workshop in Abingdon, UK, June 3 – 5, 1998.

[Car94] David A. Carr. Specification of interface interaction objects. In Beth Adelson, Susan Dumais, and Judith Olson, editors, *Proceedings of the Conference on Human Factors in Computing Systems*, pages 372–378, New York, NY, USA, April 1994. ACM Press.

[CCCL93] L.M.F. Carneiro, M.H. Coffin, D.D. Cowan, and C.J.P. Lucena. User interface high-order architectural models. Technical Report CS-93-14, Departament of Computer Science. University of Waterloo, Ontario-Canada, 1993.

[CCL93] L.M.F. Carneiro, D.D. Cowan, and C.J.P. Lucena. Advcharts: a graphical specification of abstract data views. In *Proc. Working Together CASCON'93*, pages 84–96, 1993.

[CCN97] G. Calvary, J. Coutaz, and L. Nigay. From single-user architectural design to pac*: A generic software architecture model for cscw. In *Proc. ACM Conference on Human Factors in Computing Systems CHI'97*, pages 242–249, 1997.

[Chi85] Uli H. Chi. Formal specification of user interfaces: A comparison and evaluation of four axiomatic approaches. *IEEE Transactions on Software Engineering*, 11(8):671–685, August 1985.

[Chr91] Rouff Christopher. *Specification and Rapid Prototyping of User Interfaces.* PhD thesis, University of Southern California, 1991.

[CILS93] D. Cowan, R. Ierusalimschy, C. Lucena, and T. Stepien. Abstract data views. *Structured Programming*, 14(1):1–13, 1993.

[CMN83] S.K. Card, T.P. Moran, and A. Newell. *The Psychology of Human Computer Interaction.* Lawrence Erlbaum, 1983.

[CN91] J. Coutaz and L. Nigay. Software design rules for multi-agent architectures. Deliverable 3066, Amodeus BRA, August 1991.

[Cor92] Claris Corporation. *Hypercard.* 1992.

[Cou87] J. Coutaz. Pac, an object oriented model for dialog design. In *Proc Human-Computer Interaction. Interact'87*, pages 431–436, 1987.

[Cou93] Joelle Coutaz. *The Encyclopedia of Software Engineering*, chapter Software Architecture Modelling for User Interfaces. Wiley and sons, 1993.

[DFAB98] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-Computer Interaction.* Prentice Hall, 1998.

[DH93] D. J. Duke and M. D. Harrison. Abstract interaction objects. *Computer Graphics Forum*, 12(3):25–36, 1993.

[DT99] J. Coutaz D. Thevenin. Plasticity of user interfaces: Framework and research agenda. In A. Sasse and C. Johnson, editors, *Proc. Interact99*, pages 110–117, Edinburgh, 1999. IFIP IOS Press Publ.

[Eas88] K. Eason. *Information Technology and Organizational Change.* Taylor and Francis, London, 1988.

[FDFH90] James D. Foley, Andreis van Dam, Steven Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, Reading, Massachusetts, U.S.A., 2nd edition, 1990.

[FR82] M.B. Feldman and G.T. Rogers. Toward the design and development of style-independent interactive systems. In *Proc. ACM Human Factors in Computer Systems Conference*, pages 111–116, 1982.

[Gru91] J. Grudin. CSCW introduction. *Communications of the ACM*, 34(12), December 1991.

[Hal84] D. C. Halbert. *Programming by Example.* PhD thesis, Berkeley, University of California, 1984.

[Har88] D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988.

[HB93] M.D. Harrison and P.J. Barnard. On defining requirements for interactions. In *Proc. IEEE International Workshop on Requirements Engineering*, pages 50–54, 1993.

[HB94] Donald Hearn and M. Pauline Baker. *Computer Graphics 2/E.* Prentice Hall, Englewood Cliffs, NJ., 1994.

[HC95a] A. Hussey and D. Carrington. Comparing two user-interface architectures: Mvc and pac. Technical Report 95-33, University of Queensland. Department of Computer Science, 1995.

[HC95b] A. Hussey and D. Carrington. Platform independient graphical user interface design. Technical Report 99-04, Software Verification Research Centre, School of Information Technology, The University of Queensland, Brisbane 4072, Australia, january 1995.

[HD95] M. D. Harrison and D. J. Duke. A review of formalisms for describing interactive behaviour. *Lecture Notes in Computer Science*, 896:49–75, 1995.

[Hil92] R. Hill. The abstraction-link-view paradigm: Using constraints to connect user interfaces to applications. In *Proc. ACM Conference on Human Factors in Computer Systems CHI'92*, pages 335–342, 1992.

[HSH90] H.R. Hartson, A.C. Siochi, and D. Hix. The uan: A user-oriented representation for direct manipulation interface designs. *ACM Transaction on Information Systems*, 8(3):181–203, 1990.

[IBM93] IBM. *IBM Dictionary of Computing.* McGraw-Hill, 1993.

[Inc93] Gold Disk Inc. *Astound User's Guide.* P.O. Box 789, Streetsville, Mississauga, Ontario, Canada L5M 2C2, 1993.

[ISOa] ISO. *Ergonomics Requirements for Office Work with Visual Displays Terminals: Guidance and Usability.*

[ISOb] ISO. *Software product evaluation- quality characteristics and guidelines for their use.*

[Jac83] R.J.K. Jacob. Executable specifications for a human-computer interface. In *Proc. ACM Conference on Human Factors in Computer Systems CHI'83*, pages 28–34, 1983.

[Jac86] R.J.K. Jacob. A specification language for direct manipulation user interfaces. *ACM Transactions on Graphics*, 5(4):283–317, 1986.

[KG83] R.S. Keister and G. R. Gallaway. Making software user friendly: An assessment of data entry performance. In *HFS 27th Annual Meeting Proceedings.* Human Factors Society, 1983.

[KK95] M. Kurosu and K. Kashimura. Apparent usability vs. inherent usability. In *CHI'95 Conference Companion.* Association for Computing Machinery, 1995.

[KP88] G. Krasner and S.T. Pope. A cookbook for using the model-view-controller user interface paradigm in smalltalk-80. *Journal of Object Oriented Programming*, 1(3):26–49, 1988.

[Lar92] J.A. Larson. *Interactive Software-Tools for Building Interactive User Interfaces*. Yourdon Press Computing Series, 1992.

[Lau90] B. Laurel. *The Art of Humanan Computer Interaction*. Addison-Wesley, 1990.

[LGR01] M.D. Lozano, P. González, and I. Ramos. Desarrollo y generación de interfaces de usuario a partir de técnicas de análisis de tareas y casos de uso. In *P. 2º Congreso Internacional de Interacción Persona-Ordenador*, 2001.

[Mac90] MacroMind. *MacroMind Director*. 410 Townsend, Suite 408, San Francisco, CA 94107, 1990.

[Mac96] Vijay Machiraju. A survey on research in graphical user interfaces, 1996.

[Mar92] A. Marcus. *Graphic Design for Electronic Documents and User Interfaces*. ACM Press, New York, 1992.

[Mar95] P. Markopoulos. On the Expression of Interaction Properties within an Interactor Model. In P. Palanque and R. Bastide, editors, *Design, Specification and Verification of Interactive Systems'95*, 1995.

[MC96] T. Moran and J. Carroll. *Design rationale: Concepts, techniques and use*. Lawrence Erlbaum, Mahwah, NJ, U.S.A., 1996.

[Mor38] Charles W. Morris. *Foundations of the theory of Signs*. University of Chicago Press, Chicago, U.S.A., international encyclopedia of unified science series, 1(2) edition, 1938.

[Mor81] Moran. The command language grammar: a representation for the user interface of interactive systems. *International Journal of Man-Machine Studies*, 15(1):3–50, 1981.

[Mye92] Brad A. Myers. Demostrational interfaces: A step beyond direct manipulation. *Computer*, 25(8):61–73, August 1992.

[Mye96] Brad A. Myers. *HandBook of UI Design*, chapter UIMSs, Toolkits, Interface Builders. 1996.

[Neg94] N. Negroponte. *Being Digital*. 1994.

[Ngo94] D.C.L. Ngo. *Visit: Visitor Information Systems Implementation Tool.* PhD thesis, Trinity College, Dublin, 1994.

[Nie93] J. Nielsen. *Usability Engineering.* Academic Press, London, 1993.

[NL95] W. Newman and M. Lamming. *Interactive System Design.* Addison Wesley Publishers, 1995.

[Nor83] Donald A. Norman. *Mental Models*, chapter Some observations of mental models, pages 7–14. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

[Nor88] Donald A. Norman. *The Psychology of Everyday Things.* New York: Basic Books, 1988.

[NT00] David Chek Ling Ngo and Lian Seng Teo. A mathematical theory of interface aesthetics. *Visual Mathematics: Art and Science Electronic Journal*, 2(4), 2000.

[Ols98] D. R. Olsen, Jr. *Developing User Interfaces.* Morgan Kaufmann Publishers, Inc., 1998.

[PF92] F. Paterno and G. Faconti. On the use of lotos to describe graphical interaction. In D. Diaper A. Monk and M.D. Harrison, editors, *People and Computer VII: HCI'92 Conference*, pages 155–174. Cambridge University Press, 1992.

[Pfa85] Guenther Pfaff, editor. *Proceeding of the Workshop on User Interface Management Systems held in Seeheim, FRG, November 1-3.* Springer Verlag, 1985.

[Pie31] Charles Sanders Pierce. *Collected Papers.* Harvard University Press, Cambridge, MA, 1931.

[PRS+94] Jenny Preece, Yvonne Rogers, Helen Sharp, David Benyon, Simon Holland, and Tom Carey. *Human-Computer Interaction.* Addison-Wesley Publishing, Reading, Mass., 1994.

[RGBG95] Baecker R.M., Gruding, Buxton, and Greenberg. *Human Computer Interaction: Towards the Year 2000.* Morgan Kaufman, San Francisco, 1995.

[RGP00] J. Rodeiro, M. Gea, and M. Pérez. A functional taxonomy of user interfaces. In *SIIE2000*, 2000.

[Sea93] A. Sears. Layout appropiateness: Guiding user interface design with simple task descriptions. *IEEE Transactions on Software Engineering*, 19(7):707–719, 1993.

[Sha86] B. Shackel. Ergonomics in designing for usability. In M.D. Harrison and A. Monk, editors, *People and Computers: Designing for Usability*. Cambridge University Press, 1986.

[Shi92] A. Shina. Client-server computing: Current technology review. *Communications of the ACM*, 35(7):77–98, July 1992.

[Shn83] B. Shneiderman. Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8):57–69, August 1983.

[Sil00] Paulo Pinheiro Da Silva. User interface declarative models and development environments: A survey. In *Interactive Systems: Design, Specification, and Verification, 7th International Workshop DSV-IS*, Lecture Notes in Computer Science, pages 207–226. Springer, June 2000.

[Smi97] Andy Smith. *Human Computer Factors: A Study of User and Information Systems*. McGraw-Hill, 1997.

[SPAS97] A. Savidis, A. Paramythis, D. Akoumianakis, and C. Stephanidis. Designing user-adapted interfaces: the unified design method for transformable interactions. In Gerrit van der Veer, Austin Henderson, and Susan Coles, editors, *Proceedings of the Conference on Designing Interactive Systems : Processes, Practices, Methods, and Techniques (DIS-97)*, pages 323–334, New York, August 18–20 1997. ACM Press.

[SRH85] A. Schulert, G. Rogers, and J. Hamilton. Adm-a dialogue manager. In *Proceedings SIGCHI'85: Human Factors in Computing Systems*, pages 177–183, San Francisco, CA, 1985.

[ST91] Joseph W. Sullivan and Sherman W. Tyler, editors. *Intelligent User Interfaces*. Addison-Wesley Publishing ACM Press, Reading, MA, 1991. QA 76.9 H85 A73; ACM Order number 704900.

[SW84] D.J. Streveler and A. I. Wasserman. Quantitative measures of the spatial properties of screen designs. In *INTERACT'84 Conference Proceedings*. North Holland, 1984.

[Toh98] S.C. Toh. *Cognitive and Motivational Effects of Two Multimedia Simulation Presentation Modes on Science Learning.* PhD thesis, University of Science Malaysia, Malaysia, 1998.

[Tra97] N. Tractinsky. Aesthetics and apparent usability: Empirically assessing cultural methodological issues. In *CHI'97 Conference Proceedings.* Association for Computing Machinery, 1997.

[Tul81] T.S. Tullis. An evaluation of alphanumeric, graphic and color information displays. *Human Factors*, 23:541–550, 1981.

[Tul84] T.S. Tullis. *Predicting the Usability of Alphanumeric Displays.* PhD thesis, Rice University, Kansas, 1984.

[Tul88] T.S. Tullis. *The Handbook of Human-Computer Interaction*, chapter Screen Design, pages 377–411. Elsevier Science Publishers, 1988.

[WS84] A. Wasserman and D. Shewmake. *Advances in Human-Computer Interaction, Volume 1.*, chapter The role of prototypes in the user software engineering (use) methodology. 1984.

Apéndice A

Términos utilizados

En este anexo se pretende justificar la utilización de ciertos términos en el texto, dada la cantidad de acepciones diferentes que pueden tomar. Sin embargo, cabe resaltar que la Real Academia de la Lengua Española lleva siempre un cierto retardo con respecto a términos que pueden ser de uso común sectorialmente, aunque su última versión corresponde al mes de Octubre del presente año.

A.1 Display

- comput : despliegue(m)
- put on view : exponer, presentar
- Comput : desplegar
- Display advertising. Press : Pancartas publicitarias (femenino plural)
- Display unit. Comput : monitor (m)
- Display window. Escaparate (m)

A.2 Layout

distribución(f), disposición(f), composición(f).

A.2.1 Composición

Sustantivo Femenino.

- Acción y efecto de componer.
- Obra científica, literaria o musucal.
- Imprenta: Conjunto de líneas, gareladas y planas que se reunen para la corrección de pruebas, y luego para la impresión.
- Pintura y escultura: Arte de agrupar figuras, masas y accesorios para conseguir el mejor efecto, según lo que se haya de representar.
- Naturaleza de los elementos presentes en un compuesto y proporción en que se hallan.

A.2.2 Disposición

Sustantivo Femenino.

- Acción y efecto de disponer o disponerse.
- Retórica: Ordenada colocación o distribución de las diferentes partes de una composición literaria.
- Ordenada colocación o distribución de algo.
- Lo dispuesto, establecido.
- Facultad de disponer algo.

A.2.3 Distribución

Sustantivo Femenino.

- Acción y efecto de distribuir o distribuirse.
- Retórica: Figura, especie de enumeración, en que ordenadamente se afirma o niega algo acerca de cada una de las cosas enumeradas.

A.3 Componente

Adjetivo y sustantivo masculino.

- Que compone o entra en la composición de un todo.

A.4 Apariencia

Sustantivo femenino.

- Aspecto exterior.
- Cosa que parece y no lo es.
- Filosofía: Término con que algunas filosofías aluden a aquel aspecto de las cosas que se distingue de su realidad e incluso se opone a ella.
- Aspecto exterior de una persona o cosa
- Cosa aparente (que parece)

A.5 Interfaz

Sustantivo femenino.

- Electrónica e Informática: Lugar en el que los dispositivos o sistemas se comunican o interactúan.
- Electricidad/Electrónica: Zona de comunicación o acción de un sistema sobre otro. Dispositivo que conecta dos aparatos o circuitos.
- Informática: Dispositivo capaz de transformar las señales generadas por un aparato en señales comprensibles por otro.
- Interface: Palabra inglesa con que se designa la conexión entre dos unidades; por ejemplo, componente físico destinado a enlazar entre sí dos dispositivos.

A.6 Abstracción

Sustantivo femenino.

- Acción y efecto de abstraer o abstraerse.
- En Filosofía: Operación intelectual que busca la naturaleza o esencia de una cosa a la que previamente se la ha separado de otras con las que estaba en relación.

A.7 Geometría

Sustantivo femenino.

- Parte de la matemática que estudia las propiedades y relaciones de ciertas entidades (puntos, líneas, superficies, volúmenes) que caracterizan el espacio ordinario, aunque desde un punto de vista formal y abstracto y sin atender a su eventual verificación física.

A.8 Modelo

Sustantivo masculino.

- Ejemplar o forma que uno se propone y sigue en la ejecución de una obra artística o en otra cosa.
- Representación en pequeño de alguna cosa.
- Esquema teórico, generalmente en forma matemática, de un sistema o de una realidad compleja (por ejemplo, la evolución económica de un país), que se elabora para facilitar su comprensión y el estudio de su comportamiento.
- Obra de nueva invención que se presenta como novedad y que, como tal,se registra a veces como objeto de propiedad industrial. Se clasifican en modelos de utilidad, industriales y artísticos.

- En lógica: conjunto de objetos que dan una interpretación válida a un sistema axiomático.
- Esquema teórico de un sistema o realidad compleja que se elabora para facilitar su comprensión y estudio.

A.9 Perceptible

Adjetivo.

- Que se puede comprender o percibir.

A.10 Representación

Sustantivo femenino.

- Acción y efecto de representar o representarse.
- Figura, imagen o idea que sustituye a la realidad.

A.11 Sistema

Sustantivo masculino.

- Conjunto de reglas o principios sobre una materia enlazados entre sí.
- Disposición de componentes interrelacionados para formar un todo.
- Conjunto de cosas o partes coordinadas según una ley o que, ordenadamente entre sí, contribuyen a determinado objeto o función.
- Medio o manera usados para hacer una cosa.

A.12 Topología

Sustantivo femenino.

- En análisis: Parte de la matemática que estudia el tipo de estructura inducida en un conjunto al definir entre sus elementos una serie de relaciones que corresponden a una generalización rigurosa de la idea intuitiva de proximidad.
- En análisis: en el espacio físico, la noción de proximidad está intuitivamente ligada a la de distancia. La topología viene a ser el estudio de las propiedades inherentes a la relación de proximidad, prescindiendo de la distancia. El objeto de la topología lo constituyen aquellas relaciones posicionales entre los puntos de un determinado conjunto que permanecen invariantes cuando éste se transforma de acuerdo con cierta "deformación".
- Ciencia que se dedica al estudio de los razonamientos matemáticos, prescindiendo de los significados concretos.

A.13 Visible

Adjetivo.

- Que se puede ver.
- Tan cierto y evidente que no admite duda.
- Manifiesto, evidente

A.14 Visual

Adjetivo.

- Perteneciente a la vista como instrumento o medio para ver.

Sustantivo femenino.

- Línea recta que se considera tirada desde el ojo del espectador hasta el objeto.

Apéndice B

Formato Descripción Interfaz

A continuación se realiza una recopilación de los formatos de representación utilizados en el presente trabajo.

B.1 Fichero de contenido

La extensión asignada al fichero de descripción de componentes es ".con ". Existe una definición de posición y tamaño fijos, y otra relativos.

Este fichero contiene información sobre la composición de los componentes contenido dentro de los contenedores. Se contempla la colocación de los componentes contenidos solo en el caso de una colocación múltiple, dado que si la colocación fuese individual estaría reflejada en el componente individual y no podría establecerse la relación topológica con el resto de componentes contenido. Debe tenerse en cuenta que esta relación topológica entre componentes contenido dentro de un objeto contenedor solo es necesario en el caso de una interfaz dinámica en la cual se pueden realizar operaciones visuales (como modificación de tamaño) sobre el objeto contenedor. En caso contrario con la representación de una posición y tamaño fijo es suficiente puesto que no varía su posición ni colocación.

Una descripción del formato del fichero de contenido puede encontrarse en la tabla B.1.

Item	Tipo	Descripción
COMP:	Ob	Cadena identificación comienzo composición componente
<nombre de componente>	Ob	Nombre del componente contenedor que se define
#	Ob	Separador de componente contenedor y información de su contenido
Alineado(Opciones)	Op	Los componentes contenidos están alineados
Opciones Iz(valor) De(valor) Su(valor) In(valor)		Opciones en Alineado Alineado a la izquierda a una distancia de valor píxeles Alineado a la derecha a una distancia de valor píxeles Alineado arriba a una distancia de valor píxeles Alineado abajo a una distancia de valor píxeles
Equiespaciado	Op	Los componentes contenido están equiespaciados entre sí y con los bordes del componente contenedor. Entre paréntesis se indican todos los componentes equiespaciados, separados mediante un #
(<nombres de componentes>)	Ob	Nombres de los componentes contenido
InfI(Id)	Op	El componente realiza introducción de información por parte del usuario. Se identifica la información a través de un Id
InfO(Id)	Op	El componente muestra información desde la aplicación. Se identifica la información a través de un Id

Tabla B.1: Formato del fichero de contenido de los componentes de la interfaz

B.2 Fichero de diálogo

La extensión asignada al fichero de definición de diálogo de la interfaz es ".dia". Existe una entrada por cada componente, evento que desea asociarse y el comportamiento que la interfaz debe reflejar. Este comportamiento puede afectar a las propiedades de otro componente, a las propiedades del propio componente o invocar procesos de la aplicación. Puede verse la descripción del fichero de diálogo en la tabla B.2

La desactivación de un componente implica que no responde a los eventos que tiene definidos en el fichero de diálogo. La activación es lo contrario. La propiedad de Visible = T significa que el componente está visible en la interfaz. La propiedad Visible = F significa que el componente no es visible en la interfaz, pero si tiene la propiedad Activo = T responderá a eventos que se produzcan sobre su posición y tamaño en el dispositivo de visualización. Esta propiedad no se suele utilizar, el caso más común es el de un componente invisible que tiene como razón su utilización para la colocación de objetos contenido en su interior.

La elección de estos eventos en lugar de los que se pueden emplear en las herramientas de desarrollo actuales viene dadon (como se indica en el capítulo 2), porque solo se consideran los eventos elementales de interacción de usuario, pudiendo considerarse los restantes como combinaciones o cálculos realizados sobre los elementales.

B.3 Fichero de descripción de componentes

La extensión asignada al fichero de descripción de componentes es ".des". Existe una definición de posición y tamaño fija, y otra relativa. Se distinguen tres tipos de componentes distintos a describir, y el formato varía de uno a otro debido a que ciertos atributos son particulares de cada formato. Podría optarse por un formato general que contemplase todos los atributos y utilizase solo aquellos que fuesen necesarios, pero dado que el fichero es plano (ASCII puro) se ha optado por la descripción individual. El formato de descripción de componentes gráficos generados mediante primitivas puede verse en la tabla B.3. El formato correspondiente a los componentes gráficos de texto puede verse en la tabla B.5 y el de los componentes gráficos por enumeración puede verse en la tabla B.7.

Item	Tipo	Descripción
DIAG:	Ob	Cadena identificación comienzo diálogo componente
<nombre de componente>	Ob	Nombre del componente que tiene un diálogo asociado en la interfaz
#	Ob	Separador de componente y el evento asociado
<Evento>	Ob	Evento asociado al componente
<Evento> LeftClick RighClick ReLeClick ReRiClick MouseOn Key(tecla/s) KeyPress(tecla/s) KeyRele(tecla/s)		Opciones en Evento click botón izquierdo click botón derecho Soltar el botón izquierdo Soltar el botón derecho Puntero del ratón sobre el componente Tecla pulsada y soltada Tecla pulsada y retenida Tecla soltada
#	Ob	Separador de componente y el evento asociado
<Respuesta>	Ob	Respuesta del componente dentro de la interfaz ante la ocurrencia del evento
<Respuesta> <Nombre de componente> (Activo=T/F) <Nombre de componente> (Visible=T/F) <Nombre de componente> (InfI()(T/F)) <Nombre de componente> (InfO()(T/F)) Proc(Id)		Opciones en <Respuesta> Activación o desactivación de un componente Propiedad de visible o no visible de un componente Proceso de la aplicación que se invoca ante el evento sobre el componente. La identificación del proceso se realiza a través del Id

Tabla B.2: Formato del fichero de diálogo de los componentes de la interfaz

Item	Tipo	Descripción
COMP:	Ob	Cadena identificación comienzo descripción componente
G	Ob	Carácter de identificación de componente gráfico
<nombre de componente>	Ob	Nombre del componente contenedor que se define
#	Ob	Separador
<primitivas>	Ob	Primitivas que definen el componente
<primitivas>	Ob	Opciones de primitivas
PRIM:	Ob	Cadena identificación comienzo descripción primitiva
<primitiva>	Ob	Puede tomar los valores : Rectángulo(p0, p1), línea(p0, p1), círculo(p0, radio), elipse(rectángulo(p0, p1), AngCom, AngFin), polígono(p0, p1, . . . ,pn)
#	Ob	Separador
EstiloLinea=valor	Ob	Estilo de línea de la primitiva. Por defecto continuo
#	Ob	Separador
AnchoLinea=valor	Ob	Tamaño en valor píxeles de la línea de la primitiva
#	Ob	Separador
ColorLinea=valor	Ob	Color de la línea de la primitiva especificado en RGB XXXXXX
#	Ob	Separador
ColorRelleno=valor	Ob	Color de relleno de las primitivas especificado en RGB XXXXXX
#	Ob	Separador
Posicion(OpcionesP)	Ob	Posición del componente dentro de su componente contenedor o bien dentro del dispositivo de visualización
OpcionesP		Opciones correspondientes a la posición del componente
Fija(x,y)		Posición fija del componente dentro del componente contenedor siendo (x,y) su posición superior izquierda

Tabla B.3: Formato del fichero de descripción de los componentes de la interfaz para componentes gráficos

Item	Tipo	Descripción
Relativa (OpcRel)(x,y)		Posición relativa indicada mediante restricciones topológicas, siendo (x,y) el valor inicial de posición sobre su componente contenedor
OpcRel Centrado(H,V,A) Justificado (De(valor), Iz(valor), Su(valor), In(valor)) Superior		Opciones de posición relativa Centrado del componente en su componente contenedor en Horizontal, Vertical o Ambas Justificado del componente en su componente contenedor a la Derecha, Izquierda, Superior o Inferior Su posición viene determinada por la topología indicada en el componente contenedor
#	Ob	Separador
Tamaño(OpcionesT)	Ob	Tamaño del componente dentro de su componente contenedor o bien dentro del dispositivo de visualización
OpcionesT Fijo(Tam_x,Tam_y) Relativo(Tam_x,Tam_y)		Opciones del tamaño relativo Tamaño del componente de texto fijo en su valor en los ejes x e y Tamaño del componente relativo al tamaño del componente contenedor y con valores (Tam_x,Tam_y) iniciales
#	Ob	Separador
Visible(T/F)	Ob	Propiedad de visibilidad del componente, visible en el dispositivo de visualización(T) o no(F)
# Activo(T/F)	Ob Ob	Separador Indica si el componente está activo(T) o no(F) en la interfaz. Si no está activo no responde a ningún evento
#	Op	Separador
<Entrada/Salida>	Op	Función de entrada o de salida
<Entrada/Salida> InfI()(T/F) InfO()(T/F)	Op Op Op	Descripción opciones entrada o salida Se activa la función de entrada de datos asociada al componente Se activa la función de salida de datos asociada al componente

Tabla B.4: Cont. descripción componentes gráficos

Item	Tipo	Descripción
COMP:	Ob	Cadena identificación comienzo descripción componente
T	Ob	Carácter de identificación de componente textual
<nombre de componente>	Ob	Nombre del componente contenedor que se define
#	Ob	Separador
<primitivas>	Ob	Primitivas que definen el componente de texto
#	Ob	Separador
Fuente=valor	Ob	Indicación de la fuente valor que utiliza la primitiva de texto
#	Ob	Separador
TamañoFuente=valor	Ob	Tamaño de la fuente de texto utilizada
#	Ob	Separador
#	Ob	Separador
ColorFuente=valor	Ob	Color del texto en valor con formato RGB XXXXXX
Posicion (OpcionesP)	Ob	Posición del componente de texto dentro de su componente contenedor o bien dentro del dispositivo de visualización
OpcionesP Fija(x,y) Relativa (OpcRel)(x,y)		Opciones correspondientes a la posición del componente de texto Posición fija del componente dentro del componente contenedor siendo (x,y) su posición superior izquierda Posición relativa indicada mediante restricciones topológicas, siendo (x,y) el valor inicial de posición sobre su componente contenedor
OpcRel Centrado(H,V,A) Justificado (De,Iz,Su,In)		Opciones de las posiciones relativas Centrado del componente en su componente contenedor en Horizontal, Vertical o Ambas Justificado del componente en su componente contenedor a la Derecha, Izquierda, Superior o Inferior
#	Ob	Separador

Tabla B.5: Formato del fichero de descripción de los componentes de la interfaz para componentes de texto

Item	Tipo	Descripción
Tamaño(OpcionesT)	Ob	Tamaño del componente dentro de su componente contenedor o bien dentro del dispositivo de visualización
OpcionesT Fijo(Tam_x,Tam_y) Relativo(Tam_x,Tam_y)		Opciones del tamaño relativo Tamaño del componente de texto fijo en su valor en los ejes x e y Tamaño del componente relativo al tamaño del componente contenedor y con valores (Tam_x,Tam_y) iniciales
#	Ob	Separador
Visible(T/F)	Ob	Propiedad de visibilidad del componente, en cuanto a si es visible en el dispositivo de visualización(T) o no(F)
#	Ob	Separador
Activo(T/F)	Ob	Indica si el componente está activo(T) o no(F) en la interfaz. Si no está activo no responde a ningún evento de los que tenga definidos
#	Op	Separador
<Entrada/Salida>	Op	Función de entrada o de salida
<Entrada/Salida> InfI()(T/F) InfO()(T/F)	Op Op Op	Descripción opciones entrada o salida Se activa la función de entrada de datos asociada al componente Se activa la función de salida de datos asociada al componente

Tabla B.6: Cont. descripción de los componentes de texto

Item	Tipo	Descripción
COMP:	Ob	Cadena identificación comienzo descripción componente
B	Ob	Carácter de identificación de componente por enumeración
<nombre de componente>	Ob	Nombre del componente contenedor que se define
#	Ob	Separador
<fichero>	Ob	Nombre del archivo donde se encuentra la descripción del componente por enumeración
#	Ob	Separador
Posicion (OpcionesP)	Ob	Posición del componente dentro de su componente contenedor o bien dentro del dispositivo de visualización
OpcionesP		Opciones correspondientes a la posición del componente
Fija(x,y)		Posición fija del componente dentro del componente contenedor siendo (x,y) su posición superior izquierda
Relativa (OpcRel)(x,y)		Posición relativa indicada mediante restricciones topológicas, siendo (x,y) el valor inicial de posición sobre su componente contenedor
OpcRel		Opciones de posición relativa
Centrado(H,V,A)		Centrado del componente en su componente contenedor en Horizontal, Vertical o Ambas
Justificado (De,Iz,Su,In)		Justificado del componente en su componente contenedor a la Derecha, Izquierda, Superior o Inferior
#	Ob	Separador
Tamaño(OpcionesT)	Ob	Tamaño del componente dentro de su componente contenedor o bien dentro del dispositivo de visualización

Tabla B.7: Formato del fichero de descripción de los componentes de la interfaz para componentes gráficos por enumeración

Item	Tipo	Descripción
OpcionesT		Opciones del tamaño relativo
Fijo(Tam_x,Tam_y)		Tamaño del componente de texto fijo en su valor en los ejes x e y
Relativo(Tam_x,Tam_y)		Tamaño del componente relativo al tamaño del componente contenedor y con valores (Tam_x,Tam_y) iniciales
#	Ob	Separador
Visible(T/F)	Ob	Propiedad de visibilidad del componente, en cuanto a si es visible en el dispositivo de visualización(T) o no(F)
#	Ob	Separador
Activo(T/F)	Ob	Indica si el componente esta activo(T) o no(F) en la interfaz. Si no está activo no responde a ningún evento de los que tenga definidos
#	Op	Separador
<Entrada/Salida>	Op	Función de entrada o de salida
<Entrada/Salida>	Op	Descripción opciones entrada o salida
InfI()(T/F)	Op	Se activa la función de entrada de datos asociada al componente
InfO()(T/F)	Op	Se activa la función de salida de datos asociada al componente

Tabla B.8: Cont. descripción de los componentes por enumeración

B.4 Fichero de estados de la interfaz

La extensión del fichero que refleja los estados de la interfaz es ".est". En este fichero se almacenan los posibles estados que la interfaz de usuario puede alcanzar durante su interacción con el usuario. Toda interfaz parte de un estado inicial correspondiente con el estado de inicio de interacción con el usuario, y desde el cual el usuario realiza todas sus actividades con el sistema. Una descripción de este formato puede observarse en la tabla B.9.

Para cada componente se realizará la descripción con el mismo formato que se utiliza en el fichero de descripción de componentes.

B.5 Fichero de transición entre estados

En este fichero se representan las transiciones entre estados de la interfaz, indicando cuál es el componente sobre el cual el usuario realiza la acción que provoca el cambio de estado. Existe una entrada por cada arco de unión entre estados de los existentes en el grafo de transiciones de estados. Tras cada entrada existe un retorno de carro. La extensión asignada a este fichero es ".tra" y su formato puede verse en la tabla B.10.

B.6 Fichero de tareas posibles

El fichero de tareas posibles almacena las secuencias de estados que se pueden producir entre el estado inicial y el estado final de la interfaz. Este conjunto de tareas está seleccionado de tal forma que no se repitan secuencias de comandos dentro de las tareas para evitar la influencia de los ciclos dentro del grafo. Cada entrada en el fichero corresponde con una tarea posible identificada del sistema. La extensión asignada para este fichero es ".tar" y su formato se describe en la tabla B.11.

Item	Tipo	Descripción
ESTA:	Ob	Cadena identificación comienzo descripción estado de la interfaz
ID	Ob	Identificador número secuencial del estado de la interfaz
#	Ob	Separador
<componentes>	Ob	Lista de componentes con las propiedades Visible = T ó Activo = T de entre todos los componentes de la interfaz
<componentes>		Descripción de cada componente

Tabla B.9: Formato del fichero de estados de la interfaz

Item	Tipo	Descripción
Estado	Ob	Identificador de estado origen
#	Ob	Separador
Componente	Ob	componente que provoca el cambio de estado
#	Ob	Separador
Evento	Ob	Evento que provoca el cambio de estado
#	Ob	Separador
Estado	Ob	Identificador de estado destino

Tabla B.10: Formato del fichero de transiciones entre estados

Item	Tipo	Descripción
Estado	Ob	Identificador de estado origen
(Componente#Evento)	Ob	Evento que provoca el cambio de estado y componente sobre el cual se aplica
Estado	Ob	Identificador de estado destino
(Componente#Evento)	Op	Evento que provoca el cambio de estado y componente sobre el cual se aplica
Estado	Op	Identificador de estado destino

Tabla B.11: Formato del fichero de tareas posibles