DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

UNIVERSIDAD POLITÉCNICA DE VALENCIA

P.O. Box: 22012 E-46071 Valencia (SPAIN)

Informe Técnico / Technical Report

Ref. No:13/05 **Pages: 35**
Title: Aspectos conceptuales de las interfaces en aplicaciones de gestión
Author (s): Gonzalez, Arturo

Date: 06-09-2005
Key Words:

VºBº **Author (s):**
Leader of Reasearch Group

1- Introducción

La especificación de la interfaz de usuario mantiene la relación entre dos organizaciones de componentes:

- la organización de los sucesos de usuario tal como las concibe el usuario
- la organización de las componentes de la aplicación tal como las conciben los diseñadores.

Las tareas de la especificación de la interfaz de usuario incluyen:

- La reorganización del modelo funcional del usuario o diseño global de interacciones.

  El diseño del nuevo sistema plantea una reorganización de las funciones. Cada tipo de usuario en el sistema dispondrá de un modelo de organización de las funciones que le competen. Este modelo de organización se materializa en la interfaz de usuario. Además la incorporación de un nuevo sistema soporte trae consigo nuevas funcionalidades que en el sistema original no existían. La especificación debe proporcionar un modelo de acceso o localización de los sucesos de usuario.

- La especificación de las componentes de interfaz o diseño detallado de interacciones.

  Para dar servicio a los sucesos de usuario utilizaremos componentes de interfaz como formularios, botones, estructuras de datos, etc. Será necesario especificar cada una de las componentes utilizadas y las relaciones entre ellas.

  El diseño de la capa de presentación debe hacerse considerando solo las componentes con las que el usuario interactúa. Todas las componentes que no se ofrecen al usuario, incluso aunque tengan funcionalidad relacionada con la presentación, pertenecen al diseño interno

- La especificación de las de componentes internas o diseño detallado de servicios. Cada función de usuario debe asociarse a las componentes internas que se responsabilizan de proporcionar el resultado esperado.

1.1 - Arquitectura de componentes

Las interacciones y reacciones del sistema se pueden organizar atendiendo a tres clases componentes.

CLASES DE INTERFAZ	Las clases de interfaz son las competentes para editar o mostrar estructuras de datos. Así como de las funciones vinculadas con dicha edición o navegación. Igualmente son competentes para informar de los estímulos provocados por el usuario (disparos).
CLASES DEL DOMINIO DEL PROBLEMA	Las clases del dominio del problema son las responsables de la lógica de la aplicación. Actúan como intermediario entre las estructuras de datos de la interfaz y las estructuras de datos de la BD. Son las competentes para la validación de dichas estructuras
CLASES DE SISTEMA	Las clases de Gestión del Sistema contienen funcionalidad general respecto a aspectos de sistema como puedan ser: servicios de seguridad, servicios de transacciones, servicios de comunicaciones, gestión de errores....

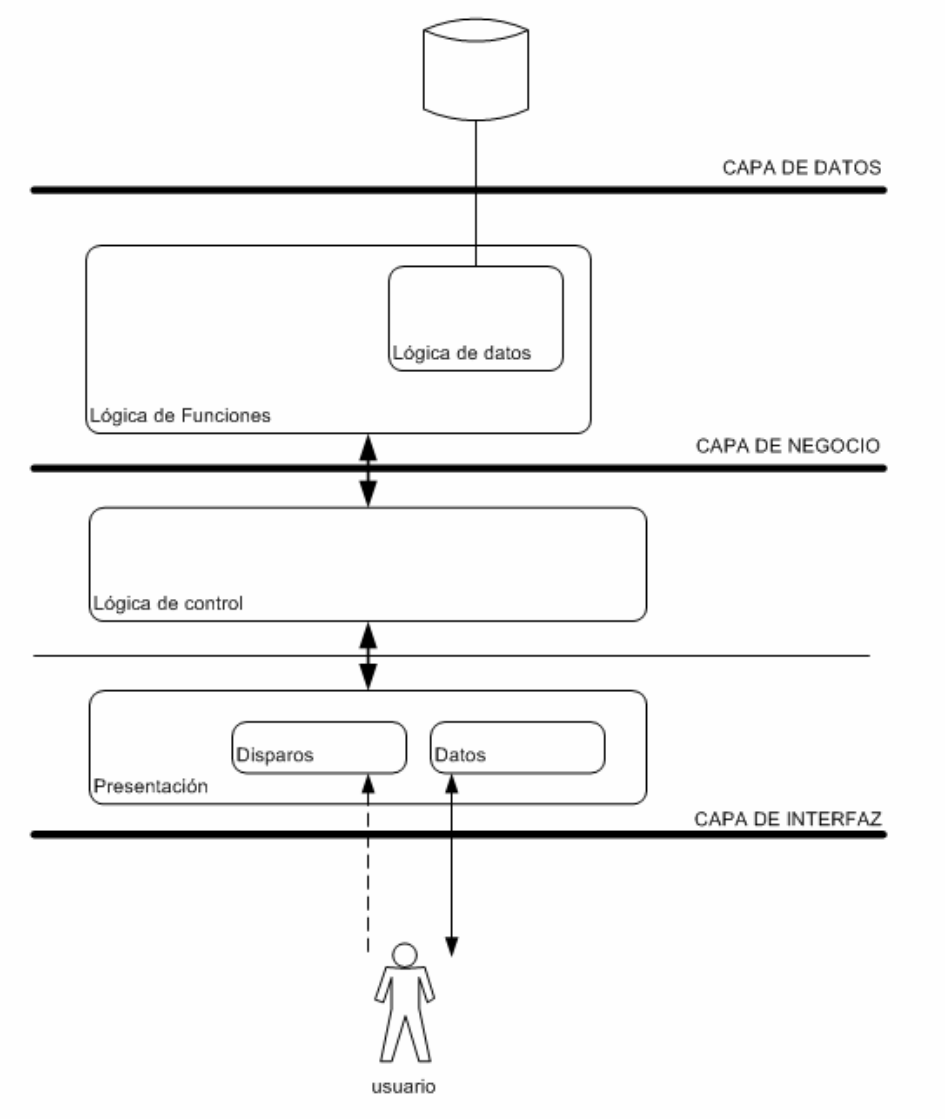
Desde el punto de vista del entorno de uso la interfaz constituye una herramienta de comunicación entre el usuario y las componentes diseñadas.

El usuario comunica señales para disparar o desencadenar la invocación de ciertas funciones preestablecidas.

El usuario comunica datos para dar forma al mensaje que quiere comunicar al sistema.

El sistema comunica datos para informar al usuario del estado de las cosas.

Estas actividades se pueden enmarcar en un entorno multicapa.



Arquitectura multicapa

1.2 - Interacciones básicas de usuario

La interfaz diseñada deberá permitir que cada usuario pueda interactuar con el sistema con las siguientes intenciones:

- Localizar la clase de función (tipo de mensaje) que quiere comunicar.
- Si la función se aplica a objetos existentes localizar la instancia de objeto al que se aplicará.
- Editar o mostrar los datos asociados con ese suceso.
- Indicar que ha finalizado la edición para disparar el servicio esperado del sistema según los requisitos del suceso comunicado.

1.2.1 - Localización.

La complejidad nos lleva a dividir y estructurar un sistema en partes relacionadas, con frecuencia de modo jerárquico. A través de las relaciones establecidas podremos **localizar** la parte que nos interese.

Desde hace tiempo se plantean dos formas básicas de organización de funciones en las interfaces de usuario. Se conocen bajo la designación de 'objetos-acciones' o 'acciones-objetos'. En cada una de ellas se da prioridad a la localización por clases de funciones o por clases de objetos.

De forma genérica podemos denominar este problema como la **localización de clases** de funciones.

Existe otro aspecto de localización en las interfaces de usuario. Se trata de la **localización de instancias**.

La localización de instancias tiene que ver con funciones que suponen la modificación o borrado de objetos existentes. Por ejemplo el cambio de domicilio de un cliente.

Pero también tiene que ver con funciones asociadas a la creación de objetos relacionados o dependientes de otros. Por ejemplo el alta de un pedido de un determinado cliente.

Estos dos tipos de localización dan lugar a dos formas de navegación que se utilizan indistintamente en una interfaz.

La primera actividad del diseño consiste en la definición de la arquitectura de invocación de sucesos de usuario. Esto se realiza mediante la **navegación explícita o taxonómica.** Es la que permite crear la organización de clases de funciones según los criterios elegidos. Se basa en el uso de enlaces o invocaciones explícitas a diferentes contextos funcionales de una aplicación. Se denomina taxonómica porque la forma de enlazar define de forma explícita la taxonomía o modelo organizativo de los sucesos de usuario.

Una jerarquía de menús es una organización de funciones basada en navegación taxonómica.

Otra actividad del diseño tiene que ver con facilitar al usuario el acceso a las instancias que debe tratar. La **navegación estructural** se basa en utilizar las relaciones del modelo de datos para facilitar la localización de instancias. Es frecuente el uso de modelos contextuales de navegación como la extensión de UML descrita en [Pastor, Abrahão et al. 2001]

Desde el punto de vista de la interfaz la navegación estructural se materializa en estructuras de interfaz asociadas a clases del modelo de datos y a sus relaciones.

1.2.2 - Edición

El proceso editorial del usuario con la interfaz le permite construir la estructura de datos que comunicará al sistema de información.

El objetivo de una interfaz además de la localización es facilitar ese proceso editorial.

Los entornos de desarrollo proporcionan componentes básicas que permiten la edición de texto alfanumérico. Son procesos de introducción de datos mediante el teclado o simplemente de introducción.

Pero en bastantes casos los datos a editar pertenecen a instancias ya conocidas por el sistema de información. Por ello se ofrecen mecanismos de navegación que permiten localizar instancias existentes de forma simple y asistida. Son procesos de introducción de datos basados en la selección de instancias existentes o simplemente de elección.

En otros casos los datos pueden ser de tipo universal, por ejemplo la fecha del día, y el propio sistema es capaz de generarlos.

Existen también datos derivados que pueden obtenerse mediante cálculos a partir de otros.

Pero además de los aspectos de edición básica existen otros que se refieren a las estructuras. Por ejemplo, indicar el inicio de edición de una estructura, indicar que se ha finalizado la edición de una o más estructuras, cancelar un proceso de edición, etc.

En el proceso de edición de los datos de un pedido el usuario tendrá una carga añadida de funciones de soporte a la edición que a veces son resultado de las limitaciones de las herramientas de desarrollo.

Estaremos editando la cabecera o las líneas. Probablemente hará falta funcionalidad para cambiar el modo de edición (modo editando cabecera, modo editando líneas).

Como la estructura detalle es un conjunto de registros incorporaremos al menos dos funciones de edición (añadir línea, eliminar línea) posiblemente tres (modificar línea).

Además tendremos que añadir las posibles funciones de edición de cada registro.

Cuando los datos del pedido sean correctos el usuario dispondrá de una función que le permita indicar que la edición se ha completado (confirmar). También será conveniente disponer de una función de cancelación de la edición.

Todas las funciones que se han presentado no son sucesos de usuario. Son funciones de edición de la interfaz.

Cuanto más compleja sea la estructura de adquisición mayor será la funcionalidad de edición.

La mayoría de entornos desarrollo permiten una relación vinculada de las estructuras de edición y las estructuras de base de datos. Es como si el proceso editorial tuviera lugar directamente sobre las estructuras de base de datos.

Esta visión vinculada ha facilitado el desarrollo de interfaces para aplicaciones de bases de datos pero en algunos casos ha forzado una visión interna o "conducida por el modelo" que ha impuesto restricciones a los usuarios. Se han asimilado los procesos editoriales y los procesos transaccionales. Buscando la simplicidad de los procesos transaccionales se ha forzado igualmente la simplicidad de los procesos editoriales.

Otro problema asociado a la vinculación de los procesos de edición y de memoria es la fragmentación de las estructuras de usuario. El usuario se ve obligado a replantear sus formularios en términos de objetos de bajo nivel: los identificadores relacionales.

1.2.3 - Disparos

La comunicación entre el usuario y la interfaz puede ser de dos tipos:

- Comunicación de datos: se suele realizar mediante estructuras capaces de editar los datos que el usuario quiere comunicar al sistema o de visualizar datos que el usuario ha solicitado al sistema.
- Comunicación de señales: mediante la cual el usuario va indicando al sistema qué desea hacer

La comunicación de control se basa en un protocolo de señales preestablecidas: los disparos o señalizaciones.

Las interfaces de ventanas permiten instrumentar la señalización de diferentes formas. Las más utilizadas son los menús y los botones de comandos.

El usuario puede comunicar una señal para disparar tres tipos distintos de funciones:

- funciones de navegación
- funciones de edición
- sucesos de usuario

2- Fundamentos de la Lógica de Presentación

En una interfaz de usuario se utilizan profusamente dos abstracciones básicas: el registro o composición de propiedades y el conjunto de registros o composición de instancias. La forma de construirlos mediante uno u otro control es irrelevante.

La abstracción elemental es el campo que tiene diferentes representaciones.

Las percepciones de las estructuras de interfaz están condicionadas por la percepción de las estructuras de memoria y se tiende a una visión disgregada en primera forma normal (1FN).

Las abstracciones básicas responden a esta percepción. Los diseñadores tienden a descomponer los formularios del usuario en términos de registros y conjuntos de registros que guardan relación con elementos de la capa de datos.

Uno de los trabajos del diseño es la fragmentación de los formularios de usuario, las estructuras de adquisición, en términos de abstracciones de interfaz conectadas por relaciones de composición o indexación.

Por ejemplo, dado el formulario de la figura inferior, el diseñador considerará en primer lugar una descomposición estructural 1FN como mínimo.

JARDICAMP
Polígono Fte. del Carro

Pedido nº 231/02
Fecha Pedido: 3-mar-02

Cliente

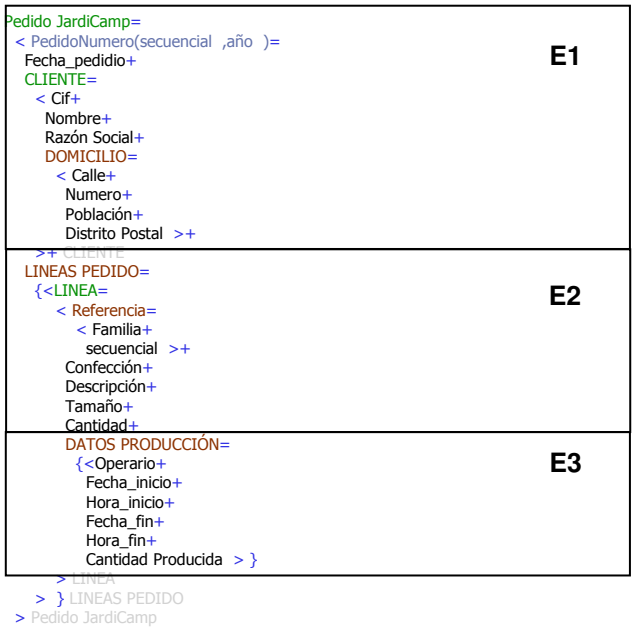
Cif	A-3204456	Razón Social	Viveros Guardiola
Nombre	José Guardiola		
Domicilio	C/ Ludovico 33-B-8	46112 Carpesa	

Ref.	Producto	T	Cant	producción	cant	producción	cant	producción	cant
F00123	Ficus reptans terracota	B	120	Luis 3/12:30 3/14.30	53	Luis 3/16:30 3/20:00	67		
F01581	Croton barro	B	14	Antonio 4/9:00 4/10:00	14				
F15001	Anemona vivero	C	400	Antonio 4/10:00 4/14:30	120	Luis 4/9:00 4/14:30	180	Antonio 4/16:30 4/20:00	100
A00253	Acebuche malla	A	20	Juan 4/10:00 4/11:30	20				

Las componentes de datos para interfaces que soportan las herramientas de desarrollo se orientan también a contenidos 1FN que mediante mecanismos de sincronización permiten composiciones complejas. Pero toda estructura sufre una descomposición de diseño.

Esta descomposición conduce a tres abstracciones estructurales relacionadas que derivan de la estructura de datos que describe el formulario.

En la figura siguiente se muestra esta fragmentación enmarcando cada una de las elecciones de estructura.



Otra decisión de diseño es el tipo de contenedores que usará para albergar las estructuras básicas.

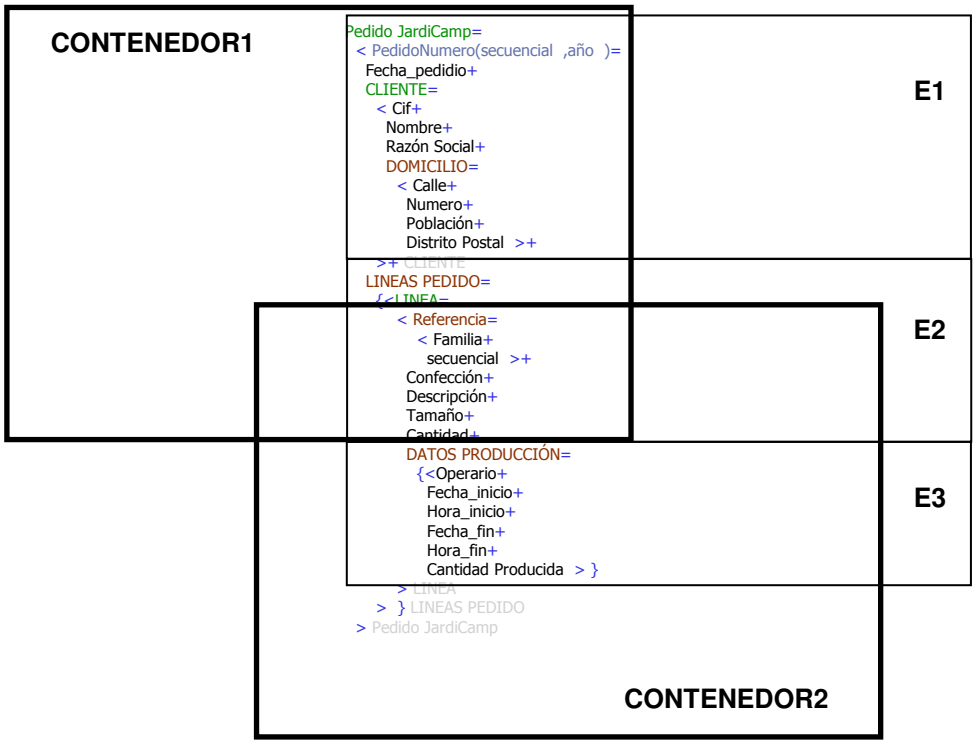
Si utiliza cliente ligero es muy probable que se utilicen varios formularios.

Si se utiliza cliente pesado es posible que todas las estructuras se acomoden en un único formulario.

De esta forma aparecen dos criterios diferentes de encapsulamiento de estructuras de interfaz:

Uno impuesto por el atavismo 1FN o de primera forma normal definida por [Codd 1970], que identifica las estructuras de la interfaz con la representación de las estructuras en la capa de datos.

Otro impuesto por el encapsulamiento de contenedor que sigue criterios de estandarización organizacional es decir de reusabilidad.



2.1 - Abstracciones de estructura de interfaz

2.1.1 - Campos elementales

Cada campo elemental de una interfaz, aparezca aislado o en el contexto de una estructura compleja, es una componente de comunicación que soporta un proceso editorial característico. Es capaz de recibir señales, que indican la cancelación o la terminación del proceso editorial, o de recibir señales para construir elementos básicos de datos de un tipo dado.

Etiquetas de campos

Describe el contenido del campo asociado.

Campos Numérico

- **Campos enteros**
  Edición/visión de valores enteros.
- **Campos decimales**
  Edición/visión de valores decimales.

Campos Temporales

- **Campos fecha**
- **Campos año**
- **Campos mes**
- **Campos hora**

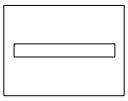
Campos de Texto

- **Línea**
- **Multilínea**
- **Formateado**

Dominios discretos o enumerados

Permite al usuario elegir entre un conjunto predefinido de valores. Esto incluye cualquier tipo de representación de la opcionalidad.

2.1.2 - Registros



La abstracción de registro se caracteriza por los campos elementales que lo constituyen y la forma de recorrerlos.

Funciones de edición:

Nuevo: presenta una estructura vacía en la interfaz.

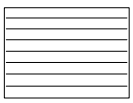
CancelarEdicion, ConfirmarEdición.

Funciones de recorrido:

CampoPrimero, CampoÚltimo, CampoAnterior,CampoSiguiente,

Habitualmente las funciones CampoAnterior y CampoSiguiente están asociadas a la tecla de tabulación.

2.1.3 - Conjuntos de registros



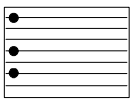
Un conjunto de registros se caracteriza por:

- tener un elemento activo (o varios si se permite selección múltiple)
- Las operaciones de navegación que permiten acceder al primero, último, siguiente o anterior registro.

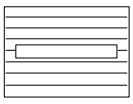
Un conjunto de registros puede tener diferentes servicios asociados. Por ejemplo servicios de búsqueda, servicios de filtrado o servicios de ordenación.

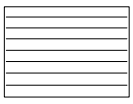
La abstracción de interfaz para el conjunto de registros puede presentar las siguientes características:

- La multiplicidad del elemento activo o aridad de la selección
  Un conjunto de registros puede tener selección unitaria en cuyo caso nos referiremos al *registro activo* del conjunto. Pero también puede tener selección múltiple.



- La multiplicidad de la visualización
  En el caso de conjuntos de selección unitaria es posible presentar solamente el elemento activo (visualización unitaria) o presentar múltiples elementos señalando el activo (visualización múltiple). En el visor unitario el usuario dispone de todas las operaciones de navegación del conjunto de registros pero solo tiene visible el elemento activo.





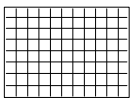
visor unitario *visor múltiple*

- La multiplicidad de la edición
  La multiplicidad de edición tiene que ver con el protocolo de confirmación de fin de edición. En el caso de edición unitaria la confirmación de edición siempre afecta a un único registro. En el caso de edición múltiple la confirmación de edición puede afectar a varios registros de la estructura.

2.1.4 - Matrices

Existe otra estructura de uso menos frecuente que las anteriores: la matriz. El control más habitual para representarla es la rejilla o *grid*.

Representaremos una estructura matricial mediante el siguiente grafismo:

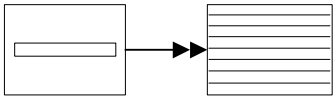


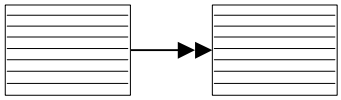
2.1.5 - Relaciones entre estructuras

Las estructuras básicas se relacionan para el diseño de una aplicación.

Relaciones de composición

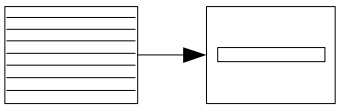
La relación maestro-detalle es una de las más conocidas. Se conoce con múltiples nombres: cabecera/líneas, padre/hijos. Fue popularizada por las bases de datos en red (set codasyl). Permite relacionar un registro con un conjunto de registros asociados que se suponen componentes, partes o hijos del registro base.



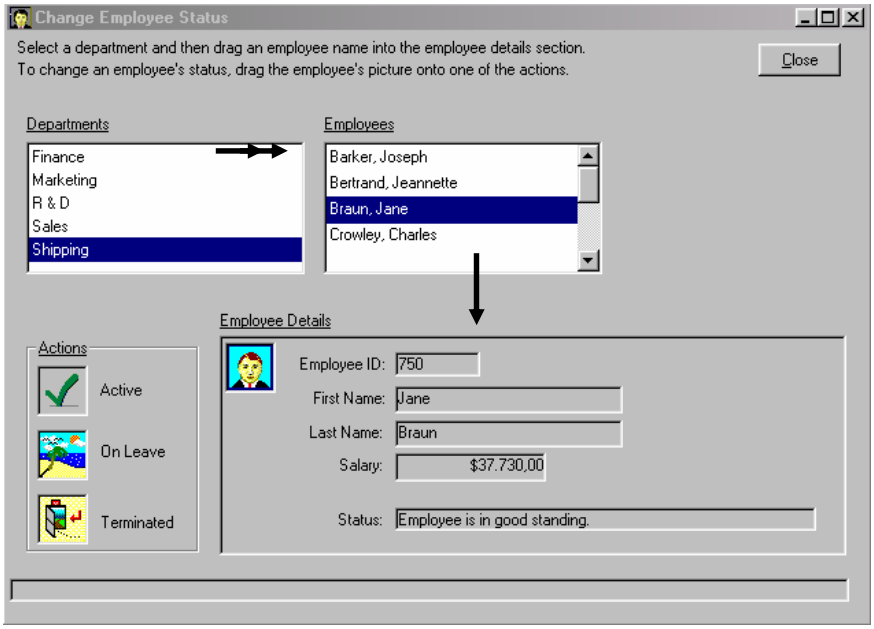


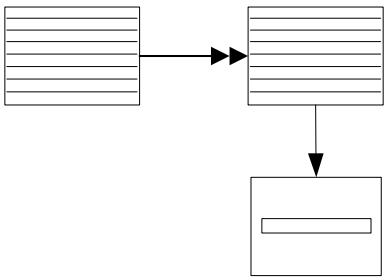
Relaciones de indexación

- Relación índice de acceso. También se conoce como relación lista/detalle.



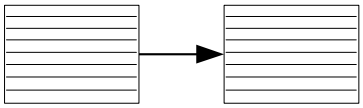
La estructura de conjunto se utiliza como un navegador que contiene información extractada y permite acceder a una estructura de registro asociada al elemento activo.





Interfaz de usuario y estructuras abstractas asociadas.

- Relación de índice sincronizado



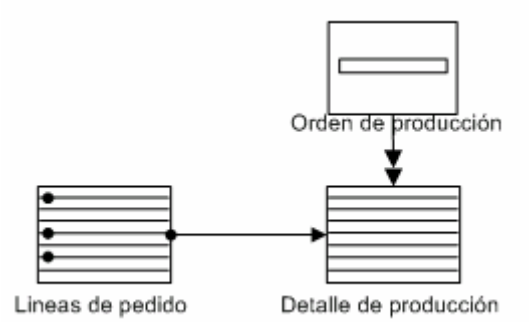
Permiten establecer una relación funcional entre dos conjuntos de modo que el elemento activo del índice se relaciona con un solo elemento del conjunto imagen.

Estos tipos de relación pueden componerse para obtener estructuras complejas de interfaz.

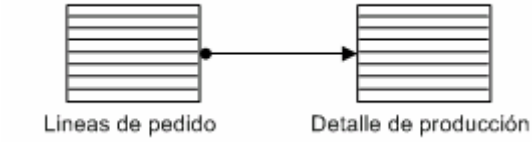
Relaciones de generación

Las relaciones generativas definen una relación constructiva entre dos estructuras de interfaz.

Supongamos que un Jefe de fabricación selecciona n líneas de pedido para asociarlas a una orden de producción. Partiendo de una estructura de conjunto de registros con multiselección, las líneas seleccionadas son el dominio origen a partir del que se construye el detalle de producción.



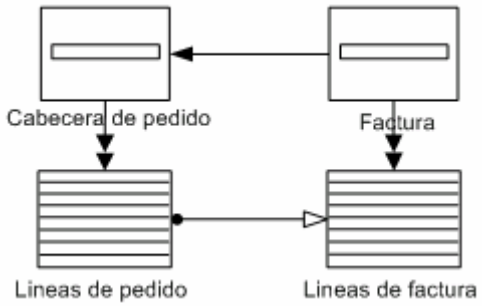
Si la estructura origen no necesita de multiselección. La relación generativa puede ser total.



Las relaciones generativas pueden ser abiertas o cerradas. Son cerradas si la estructura generada solo puede ser construida por su relación con la estructura matriz.

Son abiertas cuando la estructura generada admite alteraciones constructivas independientes de su estructura matriz.

Habitualmente las empresas facturan según pedido pero es posible que una línea del pedido no haya podido producirse por cualquier razón, o que de otra línea se hayan producido diferentes unidades o que se haya cambiado un modelo por otro.



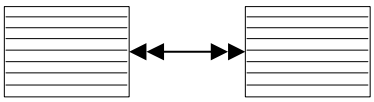
La figura anterior expresa que una cabecea de factura indexa una cabecera de pedido. Que esa cabecera de pedido se compone de múltiples líneas y que esas líneas se utilizan para generar la composición de las líneas de factura. Además esa composición puede alterarse mediante operaciones propias.

Relaciones de composición compleja

- Relación de composición múltiple

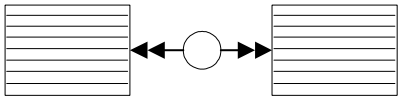
  La relación de composición múltiple tiene sentido para representar relaciones puramente asociativas en las que la participación de los elementos en la relación es simétrica y el usuario puede querer ver la relación desde los dos "lados".

  Puede ser el caso de Usuarios, Grupos y la relación Pertenece. Podemos representar el conjunto de usuarios y el conjunto de grupos. La relación se representa cada vez que accedamos a un elemento de un conjunto marcando los elementos del otro que se relacionan con él.



  La relación de composición simple es útil para relaciones agregadas en las que la relación presenta asimetría. Un pedido se compone de líneas pero una línea no se compone de pedidos.

- Relaciones composición poblacional



  La composición poblacional surge de formas de identificación basadas en las poblaciones, más que en los individuos.

  Conceptualmente conduce a formas de razonamiento diferentes de las habituales en la identificación individual. Aparecen restricciones de preservación de poblaciones o de exahustividad en la aplicación de sucesos. La composición poblacional da lugar a estados y a diagramas de transición de estado poblacionales en los que el estado viene caracterizado por la población afectada. Por ejemplo el porcentaje de cada línea de pedido que está pendiente de asignar a fabricación.

- Relaciones de composición funcionales

  Algunas relaciones de composición o generación pueden ser todavía más complejas. En la figura de la página anterior, la cabecera de factura indexa un pedido. En realidad podemos decir que contiene un selector que identifica un pedido.

  En este caso la estructura indexada, el pedido y sus líneas, se supone presente en la interfaz. Pero pueden aparecer casos donde el dominio generativo no sea tan evidente.

  El proceso de composición puede asociarse a las siguientes funcionalidades:

  Selección:

  Selectores directos: una estructura es un selector directo si contiene la información mediante la cual se seleccionarán las instancias de una clase que formará el dominio base de la generación.

  Selectores indirectos: una estructura es un selector indirecto si contiene información que permite seleccionar un conjunto de selectores.

  Proyección:

  Define las propiedades de la estructura.

  Agrupación:

Existen dos formas de agrupación, las agrupaciones de individuos y las agrupaciones de poblaciones.

En ambas podemos usar la definición de secciones de grupo habituales de los generadores de informes o la cláusula *group* by del SQL. La diferencia esta en que los grupos de individuos se muestran como complejos mientras que las agrupaciones de poblaciones se amalgaman según el criterio de población resultando en una nueva población indiferenciada.

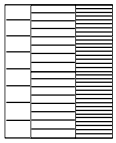
En el caso de anidación de agrupaciones no es posible el uso de una única sentencia SQL ya que el SQL solo puede contener un criterio *group by.*

Asumimos que una relación de composición se asocia a un mecanismo de clave ajena o a un criterio selectivo asociado a atributos compartidos entre dos estructuras.

¿Pero qué ocurre cuando la composición es variable en función de criterios complejos? No tenemos más remedio que recurrir a funciones.

2.1.6 - Estructuras jerárquicas multinivel

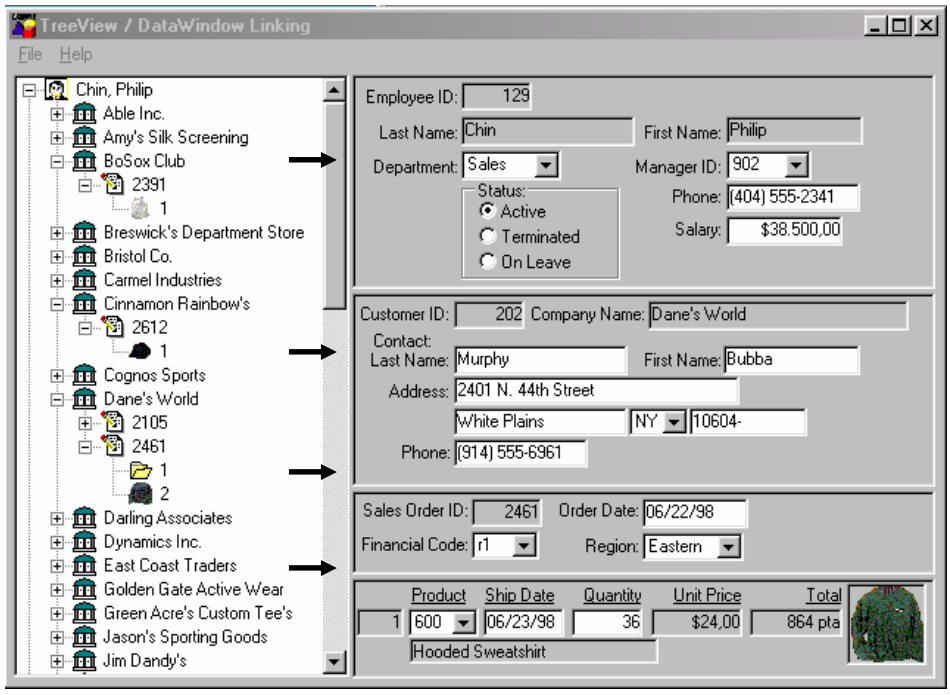
Son estructuras de interfaz que permiten una representación arborescente de relaciones 1-N entre registros. El usuario puede en cualquier momento "abrir" o "cerrar" un determinado nivel.

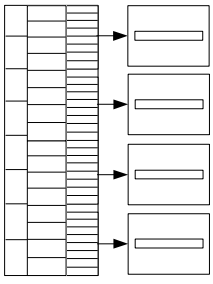


Las estructuras jerárquicas favorecen la navegación y permiten una reducción de la complejidad de componentes de interfaz para el usuario pero posiblemente suponen un incremento de las tareas de programación.

Son estructuras más complejas y sus relaciones también. Una estructura jerárquica multinivel es equivalente a un encadenamiento de relaciones de composición. Pero tiene la ventaja de mostrar selectivamente el nivel de detalle que el usuario elija.

Cada nivel de la jerarquía puede mantener relaciones diferentes con otras estructuras.





2.1.7 - Servicios

Los servicios son funcionalidades editoriales asociadas a las estructuras. Son fácilmente estandarizables por lo que se simplifica el esfuerzo de especificación.

Servicios para campos elementales

- **Servicio LUPA o ayuda de campo.**
  Permite instanciar el valor de uno o más campos mediante navegación por un conjunto de registros o campos elementales. Es un servicio para campos cuyo dominio está asociado a instancias de un modelo de datos. En la mayoría de los casos estos campos pertenecen a estructuras de tipo registro.

  El servicio de ayuda de campo se asocia a la operación de elección en las estructuras de adquisición.

  Para la especificación de ayuda de campo basta con la definición de un estándar de presentación y la estructura asociada a la elección en la forma

  ESTRUCTURA(selector)<proyección>{<visualización>}

  Donde el selector indica un posible criterio de selección sobre la clase referida, la parte de proyección indica los campos que se derivan para la interfaz y la visualización indica los campos que se presentarán al usuario para que proceda a la elección.

  Como la presentación es a fin de cuentas un conjunto de registros, siempre podrán aparecer servicios asociados.

Servicios para conjuntos de registros

- **Servicio de filtro**
  Un servicio de filtro permite que el usuario varíe las instancias asociadas a una estructura de conjunto de registros mediante criterios específicos.

  El servicio de filtro está siempre asociado a una estructura y su especificación debe estandarizarse.

  Hay dos estándares a describir:

  - El estándar de realimentación describe la forma en que se informa al usuario de la existencia de un filtro.
  - El estándar de presentación describe la forma en que el usuario podrá definir el filtro.
  - Además deberán describirse los campos específicos que se propondrán como filtro (parámetros).
- **Servicio de ordenación**
  Un servicio de ordenación permite que el usuario ordene las filas de una estructura de conjunto de registros. Está siempre asociado a una estructura y su especificación debe estandarizarse.

  Para la especificación debe describirse el estándar de definición para a forma en que el usuario pueda definir la ordenación.
- **Servicio de búsqueda**
  El servicio de búsqueda admite diferentes realizaciones. Las búsquedas más frecuentes están orientadas a carácter. Pueden realizarse sobre un campo o sobre un conjunto de campos.

  Pueden se búsquedas de subcadenas iniciales, de subcadenas en texto.

  Los servicios de búsqueda estándares deben contemplar:

  - La forma en que el usuario definirá o podrá cambiar los campos de búsqueda

- La forma en que iniciará el servicio de búsqueda, cómo se resuelve la captura del valor a buscar y la forma en que podrá proseguir la búsqueda de nuevos registros.

- **Servicio de selección**

  El servicio de selección admite dos realizaciones.

  La selección unitaria para marcar el elemento activo de un conjunto.

  La selección múltiple para permitir la definición manual o automática de subconjuntos. Estos subconjuntos permiten su uso como dominios para operaciones iteradas, para agrupaciones, etc.

Forma genérica de especificación

Servicio	Estándar	Parámetros
Nombre_del_evento	D estándar de invocación P estándar de presentación R estándar de realimentación	

Cada servicio que se use en una estructura deberá tener un nombre de disparo. A ese disparo se asociaran los posibles estándares de invocación (D) de presentación (P) y de realimentación (R).

Es posible que para un determinado servicio no haga falta o no quiera usarse realimentación por ejemplo, pues no se pone.

El uso de un sistema de filtro puede tener diferentes estándares de realimentación.

Uno puede basarse en un indicador de presencia/ausencia que al dispararlo nos lleve a la definición del filtro.

Otro puede ser mostrar siempre asociado a la estructura una frase que indique el filtro activo.

La sección de parámetros se utilizará para indicar posibles parametrizaciones de un servicio.

Por ejemplo, si usamos un servicio estándar de ordenación avanzada en el que el usuario puede elegir las columnas de ordenación, habrá que pasar como parámetro las columnas que están permitidas.

Si usamos un servicio de búsqueda, será conveniente indicar la tabla, el valor devuelto y los campos a mostrar

TABLA(devuelve){<lista de campos a mostrar>}

Los parámetros dependen de la estandarización del servicio.

2.2 - Funciones editoriales

La capa de presentación contiene funcionalidad para soportar aspectos de presentación, localización y soporte editorial.

Los servicios presentados en el apartado anterior son un ejemplo de soporte editorial.

Además la capa de presentación debe disponer de funciones para presentar las componentes (estructuras y disparos) que caracterizan y soportan un determinado proceso editorial.

2.2.1 - Localización y soporte editorial.

Una estructura de interfaz no es un mero contenedor sino que contiene funciones editoriales y de navegación estructural. Un editor/navegador es un encapsulamiento estructural dotado de funciones de localización y soporte editorial que el usuario puede provocar enviando señales a la interfaz mediante los dispositivos de captura (teclado, ratón, localizador…).

Si el editor es compuesto, el usuario puede focalizar el proceso editorial de cada una de las componentes elementales que constituyen el editor.

Los editores/navegadores se organizan en contenedores de disparadores y estructuras. Los editores más simples los denominaremos subpresentaciones. Una subpresentación es un encapsulamiento de una estructura (un registro o un conjunto de registros…) y las funciones editoriales requeridas. Una subresentación es un encapsulamiento coherente que recibe comunicaciones de señales y datos del usuario para soportar un proceso editorial específico.

2.2.2 - Funciones de Presentación

Denominaremos funciones de presentación a las funciones que organizan la representación en la interfaz de los editores navegadores es decir disparadores y estructuras.

Es habitual que una interfaz se diseñe reutilizando componentes. Los procesos editoriales de diferentes sucesos de usuario se resuelven reusando estructuras, relaciones entre estructuras y disparadores.

En tal caso las funciones de presentación tienen como responsabilidad modificar las características editoriales de la interfaz ya sea alterando las propiedades de visibilidad, accesibilidad y editabilidad de la estructura ya sea variando el repertorio de funciones editoriales adscritas.

2.3 - Reusabilidad estructural en interfaces.

Cuando se programa una interfaz es habitual el uso de un diseño modal.

En el diseño modal se definen las componentes estructurales y sobre ellas se programan diferentes modos de presentación variando sus características de editabilidad, visibilidad y apariencia o variando las funciones asociadas. De esta forma hablamos de un formulario cliente en modo alta, borrado o modificación [A|B|M].

El uso de modalidades tiene relación con la abstracción de generalización/especialización.

Existen dos formas básicas de la especialización. Se trata de la especialización característica y de la especialización adquirida.

La especialización característica está asociada a la diferenciación de propiedades entre instancias desde el momento de su creación. Las instancias pertenecen a una determinada especialidad porque son así desde que "nacen". Los cambios de especialidad, si existen, no son lo habitual.

La especialización adquirida guarda relación con los sucesos que pueden afectar a un objeto a lo largo de su vida. La especialización adquirida puede ser optativa o normativa.

Las formas normativas se asocian con diagramas de transición de estados. Por norma todo objeto debe acabar en un plazo de tiempo razonable en uno de los estados finales. El plazo de tiempo depende de la naturaleza del fenómeno modelado. Por ejemplo "vuelo en pista de despegue" puede ser una especialidad normativa de vuelo.

Las formas optativas se asocian a sucesos opcionales en la vida de un objeto. Por ejemplo estudiante es una especialización optativa de persona, que se adquiere al matricularse en un centro de estudios.

Es normativo que un vuelo acabe en tierra. Es optativo que una persona se matricule.

La típica interfaz de una estructura con los modos [A|B|M] no es sino un caso extremo de especialización optativa. El borrado y la modificación son a fin de cuentas sucesos optativos para la estructura.

2.3.1 - Generalización/especialización en sucesos.

La generalización especialización en sucesos constructores comporta:

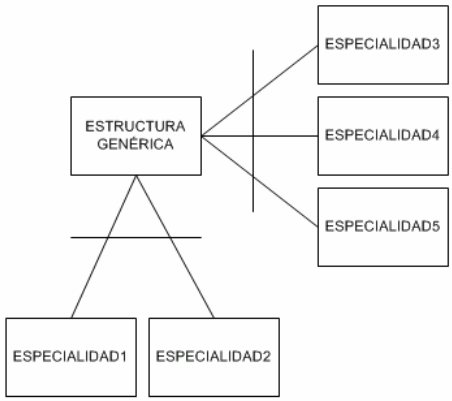
- Adquisición de datos genéricos
  La interfaz se diseñará para la adquisición de los datos requeridos por la parte genérica de la estructura
- Elección de la especialidad
  La interfaz se diseñará de modo que el usuario pueda indicar la especialidad involucrada en la actualización. Esta indicación puede tener lugar de dos formas: mediante estructuras de señalización (o estructuras basadas en tipos enumerados, el usuario elige una de las opciones que el sistema le propone.) o mediante estructuras de datos (comunicación de control mediante estructuras de datos).

  En cualquiera de los casos se trata de comunicación de control porque es información que se utiliza para disparar diferentes tratamientos. El diseñador puede utilizar una componente de control (un command button, una pestaña…) que encamine la especialidad en cuestión. O bien el usuario puede aportar un dato a partir del cual se decida el tratamiento a aplicar (un caja desplegable, una caja de texto, un radio button….)

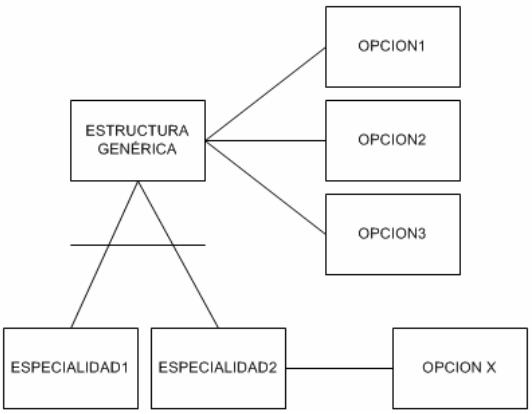
2.3.2 - Criterios para el tratamiento de la especialización generalización en el desarrollo de interfaces.

- Determinación conceptual de la jerarquía de generalización especialización.
  El desarrollador deberá establecer de forma clara la jerarquía de especialización. Primero se determinará la estructura genérica. Los datos de esta estructura serán comunes a todas las variedades así como sus características de edición y validación. La diferencia de propiedades o características aconseja la ubicación de una estructura en el nivel de jerarquía adecuado.

- Especializaciones cruzadas
  Algunas estructuras pueden presentar más de un criterio de especialización. Por ejemplo, supongamos una clase PERSONAS que puede especializarse por el tipo de ocupación (programador, jefe proyecto, administrativo…) pero además pueden especializarse por el tipo de contrato (contrato laboral fijo, contrato por servicio, contrato por horas). Si cualquier combinación es válida estamos ante una especialización cruzada donde las posibles variedades resultan de multiplicar el número de variedades de cada tipo de especialización.



- Opcionalidad
  La opcionalidad es un caso particular de especialización. La presencia en una interfaz de diferentes opcionalidades debe tratarse de modo similar a las especializaciones cruzadas. Cada opcionalidad constituiría un tipo de especialización degenerada (con una sola variedad).



Cuando se establece la jerarquía de especialización es importante asignar cada clase al nivel de emergencia que le corresponda. Así si una opción sólo se da en una determinada variedad debe adscribirse a dicha variedad.

También debe tenerse en cuenta que la jerarquía de especialización no tiene porqué presentar simetría y que una rama puede llevar a más niveles que otras.

- Especialización e interacción de usuario.
  Especialización y opcionalidad definen variedades estructurales asociadas a elecciones o condiciones establecidas por el usuario.

  Cuando el usuario "elige" una variedad estructural lo hará normalmente mediante un control de señalización es decir asociado a dominios enumerados. Las variedades estructurales de la opcionalidad y de la especialización deben establecerse en el momento del diseño conceptual y su variabilidad, por compleja que sea, debe predecirse, es decir, enumerarse.

  Las condiciones sobre campos son más sutiles. Si el campo es un enumerado (un conjunto discreto y preestablecido de valores) estamos en el primer caso. Es en los valores sobre dominios no discretos donde aparece el establecimiento de condiciones. Para que un dominio no discreto se asocie a especialización opcionalidad se debe discretizar. Debe existir alguna función que devuelva un conjunto enumerado de valores. Con frecuencia estas funciones se asocian a condiciones.

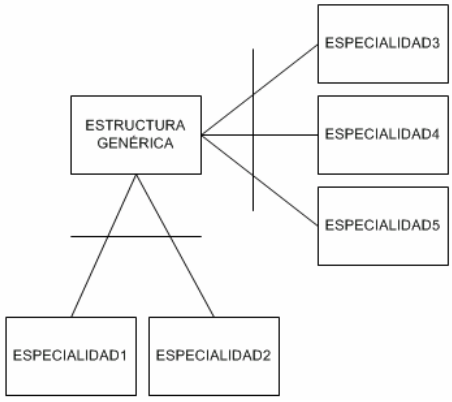
  Una especialización opcionalidad puede tener asociadas las dos formas de establecimiento. El usuario puede elegir la variante estructural y además pueden existir funciones o condiciones que validen la pertenencia a la subclase elegida. En este caso las condiciones de especialización actúan más como restricciones de clase que como selectores estructurales.

- Especialización no estructural.
  La variabilidad de clases puede afectar no solo a la variabilidad estructural sino también a la variabilidad de dominios. Es posible que dos subclases sean estructuralmente idénticas y que las condiciones de especialización establezcan subclases en los dominios de sus propiedades. Puede tratarse de formas de especialización inducida por los dominios.

2.3.3 - Recomendaciones y estándares

Cada criterio de especialización determina una colección de procedimientos (eventos/funciones/métodos) de subpresentación. Un procedimiento de subpresentación encapsula todas las características de visibilidad de, editabilidad y disparos asociados a una estructura que constituye una parte de la interfaz completa.

Las subpresentaciones están vinculadas con el tratamiento de la especialización. Bien por criterios conceptuales, el problema del usuario presenta diversidad estructural, bien por criterios de reutilización, hemos observado patrones estructurales que podemos reutilizar.



En la figura anterior existen dos criterios de especialización. Por lo tanto existirán 3 colecciones de eventos de subpresentación.

- la subpresentación genérica {S-genérica}
- la colección A de subpresentaciones {S-especialidad1, S-especialidad2}
- la colección B de subpresentaciones {S-especialidad3, S-especialidad4, S-especialidad5}

En un instante dado una presentación será una composición de la subpresentación genérica, una subpresentación de la colección A y una subpresentación de la colección B.

Desde el punto de vista de programación parece recomendable definir un procedimiento para cada colección que 'oculte' cualquier aspecto de una subpresentación previa de la colección.

Si existe una jerarquía de especializaciones compositivas la habilitación de disparos de especialización será responsabilidad de cada nivel de la jerarquía. Cada nuevo nivel de la jerarquía define una nueva colección de procedimientos de subpresentación.

2.3.4 - Compatibilidad editorial

Dados dos sucesos de usuario, diremos que son editorialmente compatibles si sus estructuras de adquisición permiten la reusabilidad estructural.

Un contexto funcional puede contener dos o más funciones editorialmente compatibles. Cuando un contexto funcional está especialmente diseñado para dar soporte a un conjunto de funciones editorialmente compatibles decimos que es un contexto editorial.

2.3.5 - Encapsulamiento de la variedad

Sea cual sea el tipo de especialización/generalización que la produzca, la reusabilidad estructural nos conduce al concepto de presentaciones o modos editoriales. Una interfaz considerada como editor puede tener una misma estructura con varios modos editoriales que modifican sus características de presentación.

Es fácil recurrir a una nominación del tipo *objeto.suceso*.

Pero cuando el objeto es complejo, es decir cuando necesitamos representarlo mediante estructuras relacionadas, cada modo o presentación del objeto nos lleva a una composición de modos de las estructuras relacionadas. Lo cual nos lleva al concepto de subpresentación o modos de estructuras básicas.

Por ejemplo, un mantenimiento de un Pedido puede llevarnos a tres presentaciones Pedido.Alta, Pedido.Consulta y Pedido.Modifcación.

Para conseguir estas tres presentaciones nos basamos en las subpresentaciones CabeceraPedido.alta, Cabecera.Pedido.Consulta, CabeceraPedido.Modificación, Lineas.consulta y Lineas.modificación.

Una presentación es un modo editorial asociado a un suceso.

Una subpresentación es un modo editorial asociado a una componente.

En el caso más simple una presentación tiene una única subpresentación. En otros casos una presentación tiene diferentes subpresentaciones.

La presentación surge habitualmente por requisitos externos o de usuario. Es el usuario el que prefiere localizar los sucesos en un mismo contexto.

Las subpresentaciones responden al problema de la adecuación de encapsulamientos para el diseño.

Lo que se debe perseguir es facilitar la modificabilidad del código y para ello priman los conceptos de localización y homogeneidad funcional.

El código de cada subpresentación no debe estar asociado a eventos de controles de forma arbitraria. Los eventos de presentaciones y subpresentaciones deben tener:

Un estándar de nominación

Un estándar de uso

Un estándar de contenidos: presentaciones y subpresentaciones tienen asociados modos editoriales que se componen de los siguientes aspectos:

- Estructura de componentes
- Características de editabilidad de los elementos de la estructura
- Funciones de soporte editorial
- Características de apariencia de la estructura
- Disparos editoriales: disparos de recorrido, de inicio de edición, de fin de edición, de cancelación de edición, de cambio de subpresentación.
- Las presentaciones tienen además asociados el disparo de función de usuario.

3- Elementos de las Lógica de Presentación: Contextos editoriales

Un contexto editorial es un conjunto relacionado de estructuras de interfaz que dan soporte a un conjunto de sucesos de usuario editorialmente compatibles.

Un contexto editorial está caracterizado por:

3.1 - Descripción general del contexto

1 Sus objetivos editoriales : descripción narrativa de los principales objetivos editoriales del contexto. En función de la complejidad del contexto puede abarcar desde varias líneas hasta varias páginas.

2 El conjunto de disparos de usuario asociados de forma genérica al contexto.

2.1 El conjunto de disparos de los sucesos de usuario de análisis a los que se da soporte editorial.

2.2 El conjunto de disparos de sucesos de usuario de diseño. Los sucesos de usuario de diseño surgen por necesidades editoriales (almacenar estados intermedios de la edición de un formulario) o por necesidades de funciones sobrevenidas de diseño (parametrizaciones de usuario, recuperaciones de históricos, ...)

2.3 El conjunto de disparos de sucesos de usuario accesibles. Los sucesos de usuario accesibles son aquellas cuyo contexto de edición se especifica separadamente pero que el usuario requiere que se localicen asociadas al contexto que se define. Estos sucesos pueden tener incluso estructuras de adquisición relacionadas con el contexto actual, pero por cualquier criterio la especificación y construcción se hace de forma separada.

Son funciones accesibles, por ejemplo, los consultores vinculados.

2.4 Funciones editoriales: Descripción de funciones editoriales relevantes ya sean disparadas por un suceso de usuario de diseño, ya sean disparadas internamente por la aplicación. Son en realidad requisitos solicitados por los usuarios pero que se entienden como objetivos secundarios.

3.2 - Descripción de los encapsulamientos editoriales.

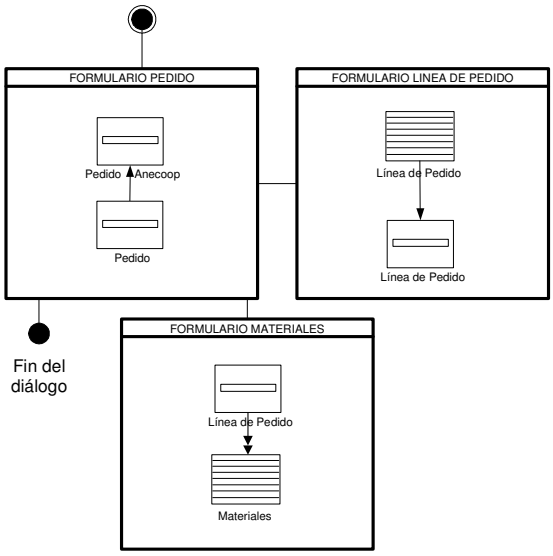
3.2.1 - Encapsulamientos multiformulario

Cuando un encapsulamiento editorial es multiformulario hablaremos de **diálogos** para referirnos a las diferentes navegaciones entre formularios que completan un encapsulamiento editorial.

A veces las estructuras de adquisición son complejas y se tiende a descomponerlas para poder tratarlas. Otras veces la descomposición no es tanto por razón de complejidad sino por criterios de estandarización de componentes.

Pero esta fragmentación editorial nunca debe imponer una fragmentación similar en la capa de negocio. En tal caso estaríamos trasladando los problemas editoriales, los problemas de la capa de presentación, a la capa de negocio.

El encapsulamiento editorial para la función de usuario alta de pedido, podría fragmentarse en los siguientes formularios.



Al no coincidir el encapsulamiento editorial con el encapsulamiento físico será necesario definirlo explícitamente. Una manera es mediante el concepto de diálogo editorial en el que presentaríamos el mapa de navegación editorial de un contexto fragmentado.

El mapa de navegación de cada diálogo deberá indicar la invocación de todos los sucesos de usuario relacionados en los apartados I.2 a I.4.

El mapa de navegación del diálogo reflejaría las interacciones externas que de forma explícita podría invocar el usuario, mediante que disparos navega de unos contenedores a otros y cuales son las formas de iniciar o finalizar el diálogo.

De existir diferentes objetivos editoriales pueden indicarse mediante agrupaciones apropiadas de formularios. Es posible que en el caso de editores complejos se añadan objetivos editoriales intermedios asociados a sucesos de diseño.

Además del mapa de navegación puede ser necesario indicar:

2- Parámetros del diálogo.

   Si el diálogo requiere parámetros se indicará su estructura y tipo.

3- Modelo de datos asociado.

   En algunos casos el modelo de datos asociado al diálogo puede ser complejo y requerir explicaciones añadidas. Por ejemplo si intervienen funciones generativas para composiciones.

Por último será necesario especificar cada uno de los formularios que componen el diálogo. Para ello se utilizarán las mismas técnicas que en los encapsulamientos mono-formulario.

3.2.2 - Formularios: Presentaciones y subpresentaciones

Una encapsulamiento editorial es una composición estructural y como tal susceptible de reutilización. La reutilización estructural de un encapsulamiento editorial conduce a formas modales.

Si la forma modal se asocia a un formulario hablaremos de presentaciones o modos del formulario. En la visión clásica [A|B|M] un formulario suele tener los modos alta, consulta, o modificación. Obsérvese que borrado no es un modo editorial sino un disparo asociado al modo consulta. No suele haber edición asociada al borrado excepto la del identificador.

Una presentación es un conjunto de estructuras relacionadas, con un repertorio de disparos asociado que se organizan en un mismo contenedor o formulario.

Cada estructura de una presentación, sus disparos asociados y sus características de editabilidad se denomina una subpresentación.

Las subpresentaciones constituyen las componentes básicas de una interfaz. La estructura de los diferentes modos es la misma o muy similar. Si hay diferencias estructurales el cambio de modo puede ocultar/presentar los campos específicos.

En cada modo pueden variar también las características de edición de algunos campos, las operaciones de captura de datos asociadas o las reglas de negocio asociadas.

La descripción de cada formulario contendrá:

1. Descripción narrativa del formulario
2. Relación de presentaciones y subpresentaciones
3. Formulario.Presentación
   3.1 Prototipo
   3.2 Mapa de composición de la presentación
   3.3 Relación entre composición editorial y modelo de datos.
4. Especificación de subpresentaciones.
   4.1 Estructura de adquisición
   4.2 Servicios
   4.3 Disparos de subpresentación
5. Disparos del formulario
6. Arquitectura de disparos del formulario

3.2.3 - Especificación de subpresentaciones

1. Estructura de adquisición

CAMPOS		Función/Parámetro	
LINEA DETALLE(LINEA PEDIDO.Campaña, LINEA PEDIDO.N°PedidoAnecoop, LINEA PEDIDO.N°Pedido, LINEA PEDIDO.N°Línea) =			
Comercial Responsable +	S		
MERCANCIA<			
Clase +	S	CLASE()[Codigo_Clase]<Nombre>	
(Variedad) +	S	VARIEDAD(Clase)[Codigo_Variedad]<Nombre>	
Categoría +	S	CLASE()[Codigo_Categoría]<Nombre>	
Calibre Máximo +	S	CALIBRE(Clase.PRODUCTO)[Codigo_Calibre]<Nombre>	
Calibre Mínimo +	S	CALIBRE(Clase.PRODUCTO)[Codigo_Calibre]<Nombre>	
Marca Pieza +	S	**FICHA PREFERENCIAS CLIENTE**	
Tratamientos S/N +	S	**FICHA PREFERENCIAS CLIENTE**	
(Tª Preenfriado) +	S	**FICHA PREFERENCIAS CLIENTE**	
>			
INFORMACION CONFECCION<			
Nº Palets +	E		RecalcularCampos(*)
Tipo Palet +	S	TIPO_PALET()[Codigo_TipoPalet]<Nombre>	RecalcularCampos(*)
Altura Cajas Paletizado +	E		RecalcularCampos(*)
Nº Cajas por Palet +	G, E	TipoPalet.PALETIZADO.CajasBase * AlturaCajasPaletizado	
Nº Cajas Totales +	E		RecalcularCampos(*)
Nº Cajas Sin Asignar Trayecto +	G	0	ActualizarMercancíaTrayecto()
Nº Palets Sin Asignar Trayecto +	G	0	ActualizarMercancíaTrayecto()
Nº Cajas Sin Asignar Cooperativa +	G	Nº Cajas Totales	
Nº Palets Sin Asignar Cooperativa +	G	Nº Palets	
Confección +	S	CONFECCION_COMERCIAL(Clase.PRODUCTO)[CodigoConfeccion]<Nombre>	
Modelo de Caja +	S	MODELO_ETRANSPORTE (CONFECCION.ETransporte)[Codigo_Clase]<Nombre>	
ET Facturable S/N +	S	MODELO_ETRANSPORTE.RetornableS/N	
Marca UVenta+	S	**FICHA PREFERENCIAS CLIENTE**	
Marca Etransporte +	S	**FICHA PREFERENCIAS CLIENTE**	
(Nº Lote Cliente) +	E		
(Código de Barras) +	S	**FICHA PREFERENCIAS CLIENTE**	
([Peso Mínimo Garantizado|Peso Pesado]) +	S		
>			
INFORMACION DE PRECIOS CLIENTE<			
(Precio Orientativo Mínimo Cliente) +	E		
(Precio Orientativo Máximo Cliente) +	G, E	Precio Orientativo Mínimo Cliente	
(Precio Acordado Cliente) +	G, E	Precio Orientativo Mínimo Cliente	

La estructura de adquisición [Gonzalez 2005] describe las características estructurales del editor. Está constituida por una estructura de datos regular que describe los campos de la presentación. Cada campo puede tener información asociada sobre:

- la operación de captura de datos del campo
- las características de visibilidad, editabilidad y obligatoriedad del campo. Un campo puede ser de uso interno y no visible para el usuario, por ejemplo un código de una línea de pedido.
- la posible vinculación del campo con la BD
- características de formato del campo
- valor inicial para el campo (se puede indicar mediante captura de generación)
- indicaciones de tipo de control a utilizar
- funciones de cálculo asociadas al campo
- etiquetas y rótulos de campo o grupo de campos.
- valores de ejemplo

La estructura de adquisición puede tener varios tipos de líneas:

- línea de selectores: describe los parámetros de selección de la estructura que permite establecer relaciones de indexación o composición con otras estructuras o con otros contextos (vía parámetros del contexto).

- LÍNEA PEDIDO(Campaña_Actual, NºPedidoAnecoop, NºPedido) =
  - Grupos de campos: agrupaciones lógicas de campos.

- Campos editables: aparecerán con una operación de captura de datos asociada.

- Campos de composición: son el resultado de concatenaciones de varios campos. Técnicamente es difícil que sean editables. Lo que no significa que el usuario no los conciba como editables.

Variedad = **CONCAT**(Clase.Abreviado, Variedad.Abreviado)	MELOC MCRE
Confección = **CONCAT**(Confección.Abreviado, ModeloCaja.Abreviado)	Alv.8k.60x40
Calibre = **CONCAT**(CalibreMínimo.Nombre, CalibreMáximo.Nombre)	1-2

2. Servicios

   Todos los servicios asociados a cada subpresentación se indicarán según se relaciona en el apartado 3.7 de este documento.

   Dada la forma genérica de los servicios las posibles realizaciones deberían estar estandarizadas tanto en el aspecto de interfaz (estándares de forma de invocarlo, usarlo o informar de su estado) como en el aspecto interno (estándares de programación, clases abstractas….)

3. Disparos

   Cada presentación o subpresentación puede tener disparos propios, ya sean disparos de edición, de cambio de presentación o disparos de sucesos de usuario.

4. Funciones de edición

   Para cada presentación o subpresentación puede ser necesaria la especificación de funciones edición asociadas. Por ejemplo funciones de derivación o cálculo de una cierta complejidad que no pueda especificarse mediante una simple expresión aritmética. También puede tratarse de funciones que se utilizan repetidamente en diferentes operaciones de captura.

   Estas funciones deberían aparecer en la descripción general del contexto.

4- Elementos de la Lógica de control

La lógica de control contiene las relaciones entre los estímulos que el usuario provoca usando los disparadores y los servicios asociados de la lógica de negocio o de la lógica de presentación.

Para representar la lógica de control utilizaremos las tablas de arquitectura de disparos de cada presentación.

Para cada formulario y presentación la arquitectura de disparos representa las respuestas del sistema a los estímulos provocados explícita o implícitamente por el usuario. Es explícito que el usuario pulse el botón aceptar. Denominaremos a un estímulo de este tipo un evento externo. Es implícito que cuando el usuario provoque un estímulo de recorrido se valide la edición en curso. El usuario no asocia explícitamente su acción con el servicio requerido. Denominaremos a los estímulos implícitos eventos internos.

Estos estímulos pueden ser de ámbito Presentación o de ámbito **subpresentación**, en cuyo caso se especificará en la primera columnna.

Las **acciones** podrán ser eventos de usuario (U) o eventos internos (I).

Los **servicios** podrán ser:

(P) Cambio de Formulario.presentación. El cambio de presentación, sea en el mismo o en diferente formulario, supone el cambio completo de entorno.

(S) Cambio de subpresentación.

Las subpresentaciones deben poder componerse para permitir el tratamiento de la especialización y de la opcionalidad.

(F) Función de negocio.

La arquitectura de disparos nunca deberá presentar descomposición funcional. Todo servicio debe asociarse a una sola función como máximo. El refinamiento de esta función pertenece al dominio de la capa de negocio. En la estructura de disparos aparecerá un nombre genérico del servicio. Posteriormente se procederá al diseño de responsabilidades imputando el servicio a un determinado objeto.

Sin embargo si puede aparecer especialización funcional si la condición de especialización está asociada a características de la capa de presentación. Por ejemplo si un campo ha sido modificado. Obsérvese que los datos contenidos no deben considerarse de la capa de presentación sino de la capa de negocio. Es decir, decidir si aplicar el tratamiento A o el B en función del valor del campo *c1* es un problema de la capa de negocio no de la capa de interfaz. Para la interfaz existe una sola función *TratamientoAoB.*

Las **condiciones** deberán indicarse como funciones. En unos casos serán funciones de interfaz *F Pedido.Cambios()* en otros funciones del sistema *f_usuario_valido( ) .*

En el ejemplo se muestra una presentación **MTTO.PEDIDOS.ALTA** que inicialmente carga las subpresentaciones asociadas a la función Alta.Inicial.

La subpresentación Línea Pedido.consulta tiene un disparo de usuario denominado Nueva Línea que provoca la invocación del servicio para habilitar la subpresentación Línea Detalle.alta .

La presentación ALTA tiene un disparo de usuario Trayectos cuyo efecto es un cambio a la presentación TRAYECTOS.Básico.

El cambio de presentación supone eliminar todas las características de la presentación actual para utilizar las características de la presentación que se invoca. Un cambio de subpresentación sólo afecta a la subpresentación del mismo nombre. Así cuando invocamos la subpresentación Línea Detalle.alta solo alteramos las características de la subpresentación línea detalle. El resto de subpresentaciones permanecen inalteradas.

MTTO. PEDIDOS	**ALTA**		
SUBPRESENTACIÓN	ACCIÓN	SERVICIO	Condiciones
	I Alta.Inicial	S Pedido Anecoop.alta S Pedido.alta S Línea Pedido.consulta S Línea Detalle.consulta S Materiales.modificación	
	U Trayectos	P TRAYECTOS.Básico	
	U Salir	[F_Guardar | F ALTA.Cerrar]	F Pedido.Cambios()
S Línea Pedido.consulta	U Nueva Línea	SLínea Detalle.alta	
	U Confirmar con Cooperativas	P CONFIRMACiÓN COOPERATIVA.Básico	
	U Borrar Línea	F Línea Pedido. Borrar	
	U Dividir Línea	F Línea Pedido .dividir	
	I LíneaPedido.CambioRegistro	S Línea Detalle.modificación	
S Línea Detalle.alta	I Línea Pedido.CambioRegistro	S Línea Detalle.Modificación	Linea_Detalle.validar ()
	U Trayectos	P TRAYECTOS.Básico	Linea_Detalle.validar ()

MTTO. PEDIDOS	**CONSULTA**		
SUBPRESENTACIÓN	ACCIÓN	SERVICIO	Condiciones
	U Confirmar con Cooperativas	P CONFIRMACiÓN COOPERATIVA .Básico	
	U Cargas por Cooperativa	P CARGAS DE VEHÍCULO POR COOPERATIVA.Básico	
	U Trayectos	P TRAYECTOS.Básico	
	U Salir		

MTTO. PEDIDOS	**MODIFICACIÓN**		
SUBPRESENTACIÓN	ACCIÓN	SERVICIO	Condiciones
	U Buscar Línea en Histórico	U HÍSTORICO LÍNEAS PEDIDO.Básico	
	F Modificación.Inicial	S Pedido Anecoop.modificación S Pedido.modificación S Línea Pedido.consulta S Línea Detalle.modificación S Materiales.Modificación	
	U Trayectos	P TRAYECTOS.Básico	f_usuario_valido( )
	U Salir	[F_Guardar | F ALTA.Cerrar]	F Pedido.Cambios()
SLínea Pedido.Consulta	U Nueva Línea	SLínea Detalle.alta	
	U Confirmar con Cooperativas	P CONFIRMACiÓN COOPERATIVA.Básico	f_usuario_valido( )
	U Borrar Línea	F Línea Pedido. borrar	f_usuario_valido( )
	U Dividir Línea	F Línea Pedido .dividir	f_usuario_valido( )
	F S Línea Pedido.CambioRegistro	SLínea Detalle.Modificación	
	F Consulta.Inicial	S Pedido Anecoop.Consulta S Pedido.Consulta S Línea Pedido.consulta S Línea Detalle.Consulta S Materiales.Consulta	

5- Especificación

LOGICA DE CONTROL

Las estructuras de disparos contienen toda la información de la lógica de control.

CAPA DE INTERFAZ

La nominación e invocación de funciones de presentación y subpresentación deberán estandarizarse.

Esta estandarización debe tener en cuenta que las presentaciones o modos podrán ser usados como parámetros.

La estandarización deberá considerar la ubicación de aspectos como la visibilidad, editabilidad, relaciones estructurales, disparos, servicios.

SEPARACIÓN DE FUNCIONALIDADES.

Todas las funciones que aparecen en la estructura de disparos deberán imputarse a la capa correspondiente.

Existen funciones en la estructura de disparos que pueden asociarse desde un principio a la capa de negocio o a la capa de presentación. Otras pueden solapar las dos capas y deberán separarse.

La función *F_guardar* asociada al evento *U Salir* contendrá una parte de funcionalidad de la capa de presentación y otra de la capa de negocio.

De la especificación de funciones de presentación, P-funciones y S-funciones, pueden derivarse invocaciones a la capa de negocio. Ya sea para derivación de datos, ya para validaciones o cálculos. Con el problema añadido de que es posible que la representación en interfaz dé lugar a código duplicado (*java-script, java-beans* …) respecto al existente en la capa de negocio (php, java…).

El acceso a la capa de negocio debe inferirse de las estructuras de adquisición a través de las expresiones selectoras.

El uso de validaciones y cálculos deberá heredarse de los requisitos funcionales. Es necesario definir la forma en que se indica qué reglas de negocio deben reproducirse en la interfaz. Probablemente se podrían clasificar : dominios enumerados, dominios de rango, dominios de clave ajena, funciones de cálculo de interfaz, funciones de cálculo con acceso a capa de datos, reglas de de especialización, reglas sobre iterados. Reglas complejas.

CAPA DE NEGOCIO

El refinamiento posterior de funciones reflejará responsabilidades de las capas de Sistema y de la capa de Datos.

Cada función de la estructura de disparos deberá refinarse para obtener una jerarquía de composición que conduzca al diseño de componentes adecuado.

En el proceso de diseño se deberán obtener:

Imputación de responsabilidades a objetos conceptuales.

Imputación de responsabilidades a objetos del sistema.

Jerarquías de abstracción de objetos de negocio y sistema.

[Codd 1970] Codd, E. F. (1970). "A relational model of data for large shared data banks." Commun. ACM **13**(6): 377-387.

[Gonzalez 2005] González, A. (2005).Estructuras de Adquisición. DSIC-II//08/05. Valencia, España. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia: 11 pp.

[Pastor, Abrahão et al. 2001] Pastor, O., S. Abrahão, et al. (2001). "Building E-commerce applications from object-oriented conceptual models." SIGecom Exch. **2**(2): 28-36.