(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 692 120**

(51) Int. CI.:

G06F 8/38 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **01.06.2005 E 05104723 (1)**

(97) Fecha y número de publicación de la concesión europea: **25.07.2018 EP 1603034**

(54) Título: **Procedimiento y aparato de generación de interfaces de usuario a base de automatización con flexibilidad total**

(30) Prioridad:

03.06.2004 US 860306

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
30.11.2018

(73) Titular/es:

MICROSOFT TECHNOLOGY LICENSING, LLC (100.0%)
One Microsoft Way
Redmond, WA 98052, US

(72) Inventor/es:

KRISTIANSEN, FREDDY;
MOLLER-PEDERSEN, JENS;
HANSEN, JESPER THEIL;
BENDSEN, PER;
CHRISTENSEN, PETER;
SLOTH, PETER;
VILLADSEN, PETER y
KJALL, UFFE

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

Aviso: En el plazo de nueve meses a contar desde la fecha de publicación en el Boletín Europeo de Patentes, de la mención de concesión de la patente europea, cualquier persona podrá oponerse ante la Oficina Europea de Patentes a la patente concedida. La oposición deberá formularse por escrito y estar motivada; sólo se considerará como formulada una vez que se haya realizado el pago de la tasa de oposición (art. 99.1 del Convenio sobre Concesión de Patentes Europeas).

DESCRIPCIÓN

Procedimiento y aparato de generación de interfaces de usuario a base de automatización con flexibilidad total

Antecedentes de la invención

La presente invención se refiere a la generación de formularios. Más particularmente, la presente invención se refiere a procedimientos y aparato de generación de interfaces de usuario (UI) de formulario.

En productos de software típicos y aplicaciones, tal como sistemas de planificación de recursos empresariales (ERP) y sistemas de gestión de relaciones con clientes (CRM), se usan un gran número de formularios o interfaces de usuario de formulario. Un formulario es una ventana, un diálogo, una página u otro elemento de UI de visualización y/o entrada de datos. No es común que el número de formularios que se usan en conjunción con una aplicación de software de negocio exceda de varios miles. Desarrollar un gran número de formularios ha sido tradicionalmente una tarea de mucho trabajo para los desarrolladores de software.

Además, está creciendo la complejidad en aplicación de negocio tal como sistemas ERP, sistemas CRM y otras aplicaciones basadas en formularios. Esto está provocado por un número de factores que incluyen: (1) un número creciente de formularios en cada sistema que se debe a funcionalidad aumentada; (2) un foco creciente sobre la usabilidad que se origina desde usuarios finales que se han acostumbrado a usabilidad de página web; (3) número creciente de diferentes plataformas, dispositivos y tecnologías; (4) un foco creciente en seguridad que puede resultar en diferentes formularios dependiendo de derechos de usuario; y (5) una demanda creciente de flexibilidad, eficiencia y personalización. Al mismo tiempo existe un empuje para desarrollar sistemas más rápidos con mayor calidad.

Como un ejemplo de una aplicación de negocio real, considérese Microsoft Business Solutions-Axapta®, que tiene cerca de 3.000 tablas, resultando en cerca de 2.000 formularios. Cada formulario tiene que alinearse con la disposición de cada tabla desde la que se vinculan los datos de tiempo de ejecución. Los formularios y lógica de formularios relacionados tienen que alinearse siempre que cambia la disposición de tabla y cuando cambia la lógica de negocio. Se añade a la complejidad el número creciente de diferentes tecnologías de plataformas cliente. La UI de Windows clásica se acompaña ahora con el explorador web. En el futuro cercano, asistente digital personal (PDA), teléfono celular y otras tecnologías de UI se añadirán a la complejidad.

La Internet ha enseñado a los usuarios finales que no necesitan un curso de 14 días para aprender cómo usar una aplicación. Los usuarios finales esperan que las aplicaciones les guíen a través de tareas y esperan que la aplicación tenga un aspecto atrayente. Porque más roles de usuarios se exponen a la tecnología de la información presentada a través de aplicaciones de negocio, existe una demanda creciente para que formularios reflejen la información que necesita cada usuario y las tareas que cada rol tiene que conseguir. En general están aumentando las demandas de experiencia de usuario.

Típicamente, la experiencia de usuario y experiencia de desarrollador se mueven en direcciones opuestas. Un desarrollador de aplicación tarda más en crear y mantener una buena experiencia de usuario. La visión de que tiene una excelente experiencia de usuario y al mismo tiempo soporta productividad de desarrollador alta, puede verse contradictorio. Esto es particularmente cierto en el área de generación formularios para aplicaciones de negocio.

Aplicaciones que presentan información deben proporcionar a sus usuarios con una experiencia tan rica como sea posible en plataformas de muy varias capacidades (oscilando desde clientes enriquecidos que ejecutan en el escritorio del usuario, a clientes Web que se ejecutan en el navegador del usuario, a Asistentes Digitales de Bolsillo, dispositivos basados en telefonía e incluso interfaces de voz). Un arquitecto de negocio usa su conocimiento en ingeniería de negocios para resolver problemas al cliente. Esta persona no es un desarrollador de programa informático y debería protegerse de las particularidades del desarrollo de programas.

La presente invención proporciona soluciones a uno o más de los problemas anteriormente mencionados y/o proporciona otras ventajas sobre la técnica anterior.

El documento US 2003/025693 se refiere a un procedimiento para proporcionar una pantalla de interfaz de usuario. Se genera un formulario electrónico independiente de una imagen de visualización objetivo. Información de control de presentación comprende sentencias declarativas para su uso en la generación de formulario de interfaz de usuario electrónico. La información de control de presentación especifica ubicaciones relativas de elementos de visualización de interfaz y elementos de aviso. Se determina la disposición general de formulario electrónico y los datos visualizados dentro del formulario. Se usa un motor de generador de códigos separado para la generación de código a partir la información de control de presentación y otros datos en un formato de código deseado. Se genera código de provisión de un formulario electrónico a base de la información de control de presentación y elementos adicionales mantenidos de forma separada. Se genera código DHTML que representa un formulario deseado a base de datos de entrada proporcionados por el constructor de interfaz de usuario que incluye información de control de presentación y artículos de datos de entrada auxiliares. Se usa información de control de presentación para dirigir la generación de código DHTML que representa una forma deseada. Información de control a su vez usa estructura de formato de datos de información formulario que contiene datos presentados en la forma deseada. Información de

control también emplea objetos que comprenden procedimientos a ejecutar en respuesta a una selección de usuario de elementos de visualización. Información de control de presentación comprende sentencias declarativas que determinan los elementos de visualización de presentación que contiene el formulario deseado. De este modo, un formulario se adapta fácilmente para diferentes tamaños de imagen de visualización y para su uso por diferentes arquitecturas de sistema.

El documento US 2003/025732 se refiere a una interfaz gráfica de usuario personalizable. Se usan ficheros de texto de XML para describir una o más interfaces de usuario. La información en el fichero de XML se usa para generar la interfaz de usuario. Actualizaciones dinámicas permiten cambios en los ficheros y la adición/eliminación de ficheros incluso en tiempo de ejecución. La disposición general de la interfaz de usuario se define en una plantilla basada en HTML. Se generan detalles de la interfaz de usuario recuperando un fichero de texto de plantilla de XML seleccionado, rellenando el texto de XML con datos de sistema actuales, convirtiendo los datos de XML a HTML y rellenando la plantilla HTML para producir un texto basado en HTML para su visualización. La interfaz de usuario se genera a continuación a partir del texto visualizado basado en HTML mediante el explorador web.

Es el objeto de la presente invención proporcionar una técnica de generación de una interfaz de usuario de formulario controlada por modelo con flexibilidad y extensibilidad mejoradas.

Este objeto se resuelve mediante el objeto de las reivindicaciones independientes.

Se definen realizaciones preferidas en las reivindicaciones dependientes.

Sumario

Se proporcionan un procedimiento, medio legible por ordenador y sistema que generan una interfaz de usuario de formulario controlada por modelo para representar una aplicación/modelo de negocio (también denominado como un modelo de datos o un modelo de dominio del problema). El procedimiento incluye seleccionar cuál de múltiples diferentes tipos de formulario lógico aplicar a un formulario lógico para generar la interfaz de usuario de formulario para representar el modelo de aplicación. El procedimiento también incluye proporcionar un primer mapa. Se genera un formulario lógico independiente del objetivo de visualización usando el modelo de aplicación, el tipo de formulario seleccionado y el primer mapa. En realizaciones de la presente invención, el primer mapa es un mapa declarativo, aunque puede tener funciones o aspectos imperativamente definidos en algunas realizaciones.

La generación del formulario lógico independiente del objetivo de visualización usando el primer mapa incluye correlacionar tipos de propiedades de dato del modelo de aplicación con controles lógicos independientes del objetivo visualización en el formulario lógico independiente del objetivo de visualización. En algunas realizaciones, el primer mapa es externo a un motor de correlación usado para generar el formulario lógico independiente del objetivo de visualización.

En algunas realizaciones, comportamientos aplicados declarativamente añaden funcionalidad al formulario lógico independiente del objetivo de visualización. Los comportamientos aplicados declarativamente se adjuntan al formulario lógico independiente del objetivo de visualización y se activan mediante eventos en el formulario. Los comportamientos declarativos pueden ser patrones de lógica que, dependiendo de valores y propiedades de los controles lógicos, establecen propiedades en otros controles.

El procedimiento puede incluir también la etapa adicional de correlación del formulario lógico con un formulario físico usando un segundo mapa. Mientras el formulario lógico contiene una pluralidad de controles lógicos, el formulario físico tiene una pluralidad de controles físicos disponibles para su uso en la presentación del formulario lógico en un objetivo de visualización. La correlación del formulario lógico al formulario físico usando el segundo mapa incluye correlacionar cada uno de los controles lógicos en el formulario lógico con uno de la pluralidad de controles físicos disponibles. Habitualmente también incluye la correlación con una disposición específica para el tipo de formulario particular en el objetivo de visualización particular.

Otras características y beneficios que caracterizan las realizaciones de la presente invención serán evidentes tras la lectura de la siguiente descripción detallada y revisión de los dibujos asociados.

Breve descripción de los dibujos

La Figura 1 es un diagrama de bloques de un entorno ilustrativo en el que puede usarse la presente invención.
La Figura 2 es un diagrama de bloques de un entorno informático móvil general en el que puede implementarse la presente invención.
La Figura 3-1 es una ilustración diagramática del uso de tipos de formulario de la presente invención para generar formularios lógicos y formularios físicos.
La Figura 3-2 es una ilustración diagramática del uso de tipos de formulario como se muestra en la Figura 3-1, y que ilustra adicionalmente la relación entre tipos de formulario y la generación de formularios lógicos y formularios físicos.
La Figura 4-1 es un diagrama de bloques que ilustra un modelo de negocio de ejemplo.
La Figura 4-2 es un diagrama de bloques que ilustra un modelo de negocio de entidad correlacionado con un

formulario.
La Figura 5-1 es un diagrama de bloques que ilustra un proceso de generación de modelos usando mapas y otros modelos.
La Figura 5-2 es un diagrama de bloques que ilustra un proceso de generación de un modelo específico de objetivo de visualización a partir de una aplicación inicial o modelo de negocio a través de una serie de correlaciones.
La Figura 5-3 es un diagrama de bloques que ilustra un proceso del tipo mostrado en las Figuras 4-1 y 4-2 para una realización de ejemplo.
La Figura 5-4 es un diagrama de bloques que ilustra un procedimiento de correlación de ejemplo en el que una entidad de modelo de negocio se correlaciona primero con un formulario independiente del objetivo de visualización, con las propiedades de entidad correlacionadas con controles para crear un formulario lógico independiente del objetivo de visualización, y a continuación el formulario lógico se correlaciona con el objetivo u objetivos de visualización
La Figura 6 es un diagrama de bloques que ilustra aspectos de tiempo de diseño y tiempo de ejecución de la presente invención y que ilustra que la capa lógica es el puente entre la lógica de negocio y el objetivo de visualización.
La Figura 7 es un diagrama de bloques que ilustra formularios lógicos correlacionados con tecnologías de presentación específicas del objetivo de visualización.
La Figura 8 es una ilustración diagramática de conceptos de la presente invención.

Descripción detallada de realizaciones ilustrativas

Con la creciente complejidad en las aplicaciones de negocios y otras aplicaciones de software administrativas o basadas en formularios, la demanda de automatización está aumentando. La presente invención proporciona un procedimiento de facilitación y mejora de tal automatización. La presente invención utiliza un enfoque declarativo y automatizado de desarrollo de interfaces de usuario sin poner en peligro la libertad de innovación.

Usando los sistemas y procedimientos de la presente invención, los desarrolladores de aplicaciones pueden centrarse en el desarrollo de un modelo de negocio. El modelo de negocio (UML, diagrama de ER, clase, etc.) se correlaciona después con un formato intermedio independiente de la tecnología que de nuevo - en una o más etapas - se correlaciona con la tecnología específica del objetivo de visualización y disposición (Windows, Navegador Web, PDA, teléfono, etc.). Los desarrolladores de marcos pueden entonces, independientemente de los desarrolladores de aplicaciones, enriquecer la tecnología específica del objetivo de visualización, y hacer que se aplique a toda la aplicación desarrollada. El formato intermedio se crea una vez por formulario y usa a través de varios objetivos de visualización. La correlación es inherentemente flexible debido a mapas abiertos y modificables usados por los motores de correlación. Los modelos de UI intermedios y los formatos de objetivo de visualización finales son también abiertos y modificables permitiendo un nivel de flexibilidad único. Las siguientes descripciones ilustran adicionalmente los conceptos inventivos.

La Figura 1 ilustra un ejemplo de un entorno 100 de sistema informático adecuado en el que puede implementarse la invención. El entorno 100 de sistema informático es únicamente un ejemplo de un entorno informático adecuado y no pretende sugerir ninguna limitación al alcance de uso o funcionalidad de la invención. Ni debería interpretarse el entorno 100 informático como que tiene alguna dependencia o requisito en relación con uno cualquiera o combinación de componentes ilustrados en el entorno 100 de operación ilustrativo.

La invención es operacional con numerosos otros entornos de sistema informático o configuraciones de fin general o fin especial. Ejemplos de sistemas informáticos bien conocidos, entornos y/o configuraciones que pueden ser adecuados para su uso con la invención incluyen, pero sin limitación, ordenadores personales, ordenadores de servidor, dispositivos de mano o portátiles, sistemas multiprocesador, sistemas basados en microprocesadores, decodificadores de salón, electrónica de consumo programable, PC de red, miniordenadores, ordenadores centrales, entornos informáticos distribuidos que incluyen cualquiera de los sistemas o dispositivos anteriores y similares.

La invención puede describirse en el contexto general de instrucciones ejecutables por ordenador, tal como módulos de programa, que se ejecutan mediante un ordenador. En general, módulos de programa incluyen rutinas, programas, objetos, componentes, estructuras de datos, etc. que realizan tareas particulares o implementan tipos de datos abstractos particulares. La invención también puede practicarse en entornos informáticos distribuidos en los que se realizan tareas mediante dispositivos de procesamiento remotos que están enlazados a través de una red de comunicaciones. En un entorno informático distribuido, pueden ubicarse módulos de programa tanto en medio de almacenamiento informático local como remoto incluyendo dispositivos de almacenamiento de memoria.

Con referencia a la Figura 1, un sistema ilustrativo de implementación de la invención incluye un dispositivo informático de fin general en forma de un ordenador 110. Componentes del ordenador 110 pueden incluir, pero sin limitación, una unidad 120 de procesamiento, una memoria 130 de sistema y un bus 121 de sistema que acopla diversos componentes de sistema incluyendo la memoria de sistema a la unidad 120 de procesamiento. El bus 121 de sistema puede ser cualquiera de varios tipos de estructuras de bus incluyendo un bus de memoria o controlador de memoria, un bus periférico y un bus local usando cualquiera de una diversidad de arquitecturas de bus. A modo de ejemplo, y no como limitación, tales arquitecturas incluyen bus de Arquitectura Estándar de la Industria (ISA), bus

de Arquitectura Micro Canal (MCA), bus de ISA mejorada (EISA), bus local de Asociación de Normalización en la Electrónica de Video (VESA) y bus de Interconexión de Componentes Periféricos (PCI) también conocido como bus de Mezzanine.

El ordenador 110 habitualmente incluye una diversidad de medios legibles por ordenador. El medio legible por ordenador puede ser cualquier medio disponible que puede accederse mediante el ordenador 110 e incluye tanto medio volátil como no volátil, memoria extraíble y no extraíble. A modo de ejemplo, y no como limitación, el medio legible por ordenador puede comprender medio de almacenamiento informático y medio de comunicación. El medio de almacenamiento informático incluye memoria tanto volátil como no volátil, extraíble como no extraíble implementada en cualquier procedimiento o tecnología para almacenamiento de información tal como instrucciones legibles por ordenador, estructuras de datos, módulos de programa u otros datos. El medio de almacenamiento informático incluye, pero sin limitación, RAM, ROM, EEPROM, memoria flash u otra tecnología de memoria, CD-ROM, discos versátiles digitales (DVD) u otro almacenamiento de disco óptico, cintas magnéticas, cinta magnética, almacenamiento de disco magnético u otros dispositivos de almacenamiento magnético, o cualquier otro medio que puede usarse para almacenar la información deseada y que puede accederse mediante el ordenador 110. El medio de comunicación habitualmente incorpora instrucciones legibles por ordenador, estructuras de datos, módulos de programa u otros datos en una señal de datos modulada tal como una onda portadora u otro mecanismo de transporte e incluye cualquier medio de distribución de información. La expresión "señal de datos modulada" significa una señal que tiene una o más de sus características establecidas o cambiadas de tal manera que codifica información en la señal. A modo de ejemplo, y no como limitación, el medio de comunicación incluye medio por cable tal como una red por cable o conexión por cable directo, y medio inalámbrico tal como acústico, RF, infrarrojos y otro medio inalámbrico. Combinaciones de cualquiera de los anteriores también debería incluirse dentro del alcance de medio legible por ordenador.

La memoria 130 de sistema incluye medio de almacenamiento informático en forma de memoria volátil y/o no volátil tal como memoria 131 de sólo lectura (ROM) y memoria 132 de acceso aleatorio (RAM). Un sistema 133 básico de entrada/salida (BIOS), que contiene las rutinas básicas que ayudan a transferir información entre elementos dentro del ordenador 110, tal como durante el inicio, se almacena habitualmente en la ROM 131. La RAM 132 habitualmente contiene datos y/o módulos de programa que son accesibles inmediatamente a y/o en los que se operan actualmente mediante la unidad 120 de procesamiento. A modo de ejemplo, y no como limitación, la Figura 1 ilustra el sistema 134 operativo, programas 135 de aplicación, otros módulos 136 de programa y datos 137 de programa. Un grupo particular de programas de aplicación se llaman aplicaciones de negocio. Estas se dirigen a la gestión de empresas incluyendo - pero sin limitación - tratar el libro de contabilidad general, inventario, salarios, clientes, ventas, compras, informes financieros y cualquier otro dato relevante para un negocio.

El ordenador 110 también puede incluir otro medio de almacenamiento informático extraíble/no extraíble y volátil/no volátil. A modo de ejemplo únicamente, la Figura 1 ilustra una unidad 141 de disco duro que lee de o escribe a medio magnético no extraíble y no volátil, una unidad 151 de disco magnético que lee de o escribe a un disco 152 magnético extraíble y no volátil, y una unidad 155 de disco óptico que lee de o escribe a un disco 156 óptico extraíble y no volátil tal como un CD ROM u otro medio óptico. Otro medio de almacenamiento informático extraíble/no extraíble y volátil/no volátil que puede usarse en el entorno de operación ilustrativo incluye, pero sin limitación, casetes de cinta magnética, tarjetas de memoria flash, discos versátiles digitales, cinta de video digital, RAM de estado sólido, ROM de estado sólido y similares. La unidad 141 de disco duro se conecta habitualmente al bus 121 de sistema a través de a interfaz de memoria no extraíble tal como la interfaz 140, y la unidad 151 de disco magnético y unidad 155 de disco óptico se conectan habitualmente al bus 121 de sistema mediante una interfaz de memoria extraíble, tal como la interfaz 150.

Las unidades de disco y sus medios de almacenamiento informático asociados analizados anteriormente e ilustrados en la Figura 1, proporcionan almacenamiento de instrucciones legibles por ordenador, estructuras de datos, módulos de programa y otros datos para el ordenador 110. En la Figura 1, por ejemplo, la unidad 141 de disco duro se ilustra como que almacena el sistema 144 operativo, programas 145 de aplicación, otros módulos 146 de programa y datos 147 de programa. Obsérvese que estos componentes pueden o bien ser los mismos que o diferentes del sistema 134 operativo, programas 135 de aplicación, otros módulos 136 de programa y datos 137 de programa. Se proporcionan diferentes números al sistema 144 operativo, programas 145 de aplicación, otros módulos 146 de programa y datos 147 de programa en este punto para ilustrar que, como mínimo, son copias diferentes.

Un usuario puede introducir órdenes e información en el ordenador 110 a través de dispositivos de entrada tal como un teclado 162, un micrófono 163 y un dispositivo 161 apuntador, tal como un ratón, bola de mando o panel táctil. Otros dispositivos de entrada (no mostrados) pueden incluir una palanca de mandos, control de juegos, antena parabólica, escáner o similar. Estos y otros dispositivos de entrada se conectan a menudo a la unidad 120 de procesamiento a través de una interfaz 160 de entrada de usuario que se acopla al bus de sistema, pero pueden conectarse mediante otra interfaz y estructuras de bus, tal como un puerto paralelo, puerto de juegos o un bus serial universal (USB). Los dispositivos de entrada se usan para la creación, modificación y borrado de datos. Dispositivos de entrada también pueden usarse para el control (inicio y parada) de los programas de aplicación y funciones particulares en el presente documento. Las funciones incluyen formularios de apertura (muestra) y cierre de los formularios. Un monitor 191 u otro tipo de dispositivo de visualización también se conecta al bus 121 de sistema a través de una interfaz, tal como una interfaz 190 de video. Además del monitor, los ordenadores también pueden

incluir otros dispositivos de salida periféricos tal como altavoces 197 e impresora 196, que puede conectarse a través de una interfaz 195 periférica de salida. El monitor u otro dispositivo de visualización se usa para mostrar (presentar) formularios.

El ordenador 110 puede operar en un entorno en red usando conexiones lógicas a uno o más ordenadores remotos, tal como un ordenador 180 remoto. El ordenador 180 remoto puede ser un ordenador personal, un dispositivo de mano, un servidor, un encaminador, un PC de red, un dispositivo de par u otro nodo de red común, y habitualmente incluye muchos o todos los elementos descritos anteriormente en relación al ordenador 110. Las conexiones lógicas representadas en la Figura 1 incluyen una red 171 de área local (LAN) y una red 173 de área extensa (WAN), pero también pueden incluir otras redes. Tales entornos de red son comunes en oficinas, redes informáticas de empresas, intranets y la Internet.

Cuando se usa en un entorno de interconexión de red LAN, el ordenador 110 se conecta a la LAN 171 a través de una interfaz de red o adaptador 170. Cuando se usa en un entorno de interconexión de red WAN, el ordenador 110 habitualmente incluye un módem 172 u otro medio de establecimiento de comunicaciones a través de la WAN 173, tal como internet. El módem 172, que puede ser interno o externo, puede conectarse al bus 121 de sistema a través de la interfaz 160 de entrada de usuario, u otro mecanismo apropiado. En un entorno en red, módulos de programa representados con relación al ordenador 110, o porciones de los mismos, pueden almacenarse en el dispositivo de almacenamiento de memoria remoto. A modo de ejemplo, y no como limitación, la Figura 1 ilustra programas 185 de aplicación remotos como que residen en el ordenador 180 remoto. Se apreciará que las conexiones de red mostradas son ilustrativas y pueden usarse otros medios de establecimiento de un enlace de comunicaciones entre los ordenadores.

La Figura 2 es un diagrama de bloques de un dispositivo 200 móvil, que es un entorno informático ilustrativo alternativo. El dispositivo 200 móvil incluye un microprocesador 202, memoria 204, componentes 206 de entrada/salida (I/O) y una interfaz 208 de comunicación de comunicación con ordenadores remotos u otros dispositivos móviles. En una realización, los componentes anteriormente mencionados se acoplan para comunicación entre sí a través de un bus 210 adecuado.

La memoria 204 se implementa como memoria electrónica no volátil tal como memoria de acceso aleatorio (RAM) con un módulo de respaldo de batería (no mostrado) de tal forma que la información almacenada en la memoria 204 no se pierde cuando se corta la potencia general al dispositivo 200 móvil. Una porción de la memoria 204 se asigna preferentemente como memoria direccionable para ejecución de programa, mientras otra porción de la memoria 204 se usa preferentemente para almacenamiento, tal como para simular almacenamiento en una unidad de disco.

La memoria 204 incluye un sistema 212 operativo, programas 214 de aplicación así como un almacén 216 de objetos. Durante operación, el sistema 212 operativo se ejecuta preferentemente mediante el procesador 202 de la memoria 204. El sistema 212 operativo, en una realización preferida, es un sistema operativo de la marca WINDOWS ® CE comercialmente disponible en Microsoft Corporation. El sistema 212 operativo se diseña preferentemente para dispositivos móviles e implementa características de base de datos que pueden utilizarse mediante las aplicaciones 214 a través de un conjunto de interfaces de programación de aplicación y procedimientos expuestos. Los objetos en el almacén 216 de objetos se mantienen mediante las aplicaciones 214 y sistema 212 operativo, al menos parcialmente en respuesta a llamadas a las interfaces de programación de aplicación y procedimientos expuestos.

La interfaz 208 de comunicación representa numerosos dispositivos y tecnologías que permiten que el dispositivo 200 móvil envíe y reciba información. Los dispositivos incluyen módems por cable e inalámbricos, receptores por satélite y sintonizadores de radiodifusión por nombrar algunos. El dispositivo 200 móvil también puede conectarse directamente a un ordenador para intercambiar datos con el mismo. En tales casos, la interfaz 208 de comunicación puede ser un transceptor de infrarrojos o una conexión de comunicación en serie o paralelo, todas las cuales son capaces de transmitir información de difusión en continuo.

Los componentes 206 de entrada/salida incluyen una diversidad de dispositivos de entrada tal como una pantalla táctil, botones, ruedas y un micrófono así como una diversidad de dispositivos de salida que incluyen un generador de audio, un dispositivo de vibración y una pantalla. Los dispositivos listados anteriormente son a modo de ejemplo y no necesitan estar todos presentes en el dispositivo 200 móvil. Además, pueden adjuntarse otros dispositivos de entrada/salida a o encontrarse con el dispositivo 200 móvil.

ARQUITECTURA DE UI CORRELACIONADA Y POR CAPAS Y PROCEDIMIENTO

Como se ha descrito anteriormente, aplicaciones que presentan información deben proporcionar a usuarios con una experiencia tan rica como sea posible en plataformas (por ejemplo objetivos de visualización) de muchas varias capacidades. Estas plataformas abarcan desde clientes enriquecidos que se ejecutan en el escritorio del usuario, a clientes Web que se ejecutan en el navegador del usuario, a PDA, a dispositivos basados en telefonía e incluso interfaces de voz. También son posibles otras plataformas. De acuerdo con realizaciones de la presente invención, se emplea un esquema que define perceptivamente cómo los tipos de datos se correlacionan con controles nativos en la plataforma en cuestión.

Un arquitecto de negocio usa su conocimiento en ingeniería de negocios para resolver problemas a sus clientes. Sin embargo, esta persona habitualmente no es un desarrollador de programa informático e idealmente se protege de las particularidades del desarrollo de programas. La presente invención proporciona procedimientos y aparato que permiten que un arquitecto de negocio (usuario) se centre en la lógica de negocio de una aplicación y no en cómo se presentan los datos en una plataforma dada. La invención presentada permite que los desarrolladores de aplicaciones, a base de un modelo del negocio, obtengan la interfaz de usuario que se dirige a múltiples objetivos de visualización y que explota las capacidades tecnológicas de cada objetivo de visualización.

Esto se consigue mediante una definición de UI por capas que permite que el desarrollador centre el esfuerzo intelectual a la capa que tiene el nivel de abstracción "correcto". Se captura y reutilizan patrones en cada nivel de abstracción. Mapas - desde niveles superiores de abstracción a niveles más específicos - capturan patrones en el procedimiento de correlación, pero también permiten al desarrollador de aplicación afinar los resultados. La presente invención usa estos mapas, junto con controles, tipos de formularios lógicos y comportamientos, para correlacionar el modelo de negocio con modelos físicos específicos de objetivo de visualización.

En los procedimientos de la presente invención, el modelo de negocio es el maestro. El trabajo intelectual puesto en el modelo de negocio se preserva y usa como base de generación de la interfaz de usuario. El modelo puede describirse en Lenguaje de Modelización Unificado (UML), diagrama de recursos de empresa (ER) o cualquier otro lenguaje de modelización gráfico o no gráfico. Modelos de negocio también pueden encontrarse en programas orientados a objeto clásicos, en bases de datos relacionales o en otros formatos. Como se describe a continuación, en algunas realizaciones ilustrativas de la presente invención, el modelo de negocio es una jerarquía de clases de entidades con propiedades.

En realizaciones de la presente invención, el modelo de negocio se correlaciona con un modelo intermedio de la interfaz de usuario que es independiente del objetivo de visualización. El modelo de UI se describe en un alto nivel de abstracción que permite al desarrollador de aplicación ver y modificar la misma sin la necesidad de ningún conocimiento técnico del objetivo de visualización final. Puede añadirse código al modelo de UI que trabajará para todos los objetivos de visualización. Patrones en los modelos de UI se capturan y pueden reutilizarse, resultando en una UI homogénea. Los componentes básicos usados para describir el modelo de UI también pueden extenderse.

Como se describirán a continuación en mayor detalle, en algunas realizaciones de la invención las entidades se correlacionan con un modelo de UI llamado "la UI lógica" que incluye un formulario lógico con controles lógicos. Las entidades y propiedades se correlacionan con formularios lógicos y controles lógicos respectivamente en tiempo de diseño. En tiempo de ejecución los formularios lógicos y datos de controles lógicos se unen a las entidades y propiedades. Los patrones en los formularios se llaman tipos de formulario. Los formularios y controles se usan mediante, pero no son una a parte de, el motor de correlación de cliente principal. El cliente principal es la parte independiente del objetivo de visualización del cliente de UI. En realizaciones de la invención, la lista de controles lógicos y tipos de formulario puede extenderse. A continuación también se proporciona una descripción adicional de estos aspectos de la invención.

En algunas realizaciones de la invención, el modelo de UI se correlaciona con un lenguaje de marcas específico de objetivo de visualización tal como el lenguaje de páginas de servidor activo (ASP.NET), el lenguaje de marcas inalámbrico (WML), el lenguaje de marcas de hipertexto (HTML), etc. El modelo de UI también puede correlacionarse con otras tecnologías de visualización (por ejemplo, Win32, WinForms, PocketPC). En un ejemplo de realización particular de la invención, los formularios lógicos se correlacionan con un objetivo de visualización de Cliente Web, un objetivo de visualización de Cliente de WinForms, etc., y pueden añadirse nuevos objetivos de visualización dinámicamente.

De acuerdo con realizaciones de la invención, las correlaciones se basan en mapas declarativos que son abiertos y modificables. Patrones en la UI se capturan en los mapas. Los mapas pueden hacerse a base de varias condiciones lo que resulta en un nivel alto de flexibilidad. Los diferentes modelos pueden modificarse manualmente (incluyendo a través de código) después de que se han correlacionado. Esto proporciona al desarrollador el nivel final de flexibilidad. La naturaleza dinámica de los mapas permite el desarrollo y uso de nuevos elementos de UI tal como controles. Los nuevos controles simplemente se habilitan en un formulario, correlacionando propiedades a los nuevos controles. Los nuevos controles deben cumplir con un cierto contrato - en esta realización los nuevos controles deben heredar de una clase base especificada. En tiempo de ejecución, el formulario lógico se une a las entidades y el objetivo de visualización se une a la capa lógica. Esto permite que toda funcionalidad que es común entre los objetivos de visualización (seguridad, personalización, lógica dentro de formularios) se implemente una vez en la capa lógica y use en muchos casos.

TIPOS DE FORMULARIO

La presente invención utiliza los conceptos de formularios lógicos y tipos de formulario lógico para proporcionar un nuevo procedimiento de construcción de interfaces de usuario de formulario (formularios) para negocios y otras aplicaciones. Mientras en realizaciones ilustrativas o implementaciones de la invención se usan formularios lógicos y tipos de formulario lógico, no se requiere el uso de la capa lógica en todas las realizaciones. Por lo tanto, la presente invención se aplica al uso de tipos de formulario para crear formularios en general. Aplicaciones de negocio de hoy

habitualmente que consiste en un gran número de formularios que a menudo se encuadran dentro de unas pocas categorías o siguen un patrón similar. El número de categorías es habitualmente entre dos y veinte, pero pueden definirse más. Los tipos de formulario de la presente invención facilitan interfaces de usuario controladas por modelos preservando y actuando en el modelo de negocio o de aplicación o ambos en tiempo de diseño y en tiempo de ejecución. Esto proporciona un nivel alto de abstracción a un desarrollador de software. Además, el uso de tipos de formulario de acuerdo con la presente invención garantiza la reutilización (una disposición se usa muchas veces), un conjunto mucho más homogéneo de interfaces de usuario de formulario ya que todos los formularios se encuadran dentro de unos pocos tipos distintos y formularios que son más fáciles de mantener (la disposición del tipo puede cambiarse sin cambiar el formulario y puede aplicarse un tipo diferente sin cambiar el formulario). La invención garantiza que desarrolladores de aplicaciones tienen control completo, a través de múltiples objetivos de visualización, del aspecto y comportamiento de la aplicación y cómo se efectúa la navegación dentro de la aplicación. Ejemplos de objetivos de visualización incluyen cada uno de los múltiples tipos de sistemas operativos actuales y futuros, así como cada uno de los muchos dispositivos móviles disponibles o futuros. Como otro ejemplo, cada tecnología de presentación en un sistema operativo particular también puede ser un objetivo de visualización.

Usando lo conceptos de la presente invención, un formulario lógico contiene controles lógicos independientes de objetivo de visualización, que hace el formulario lógico independiente de los propios objetivos de visualización. Un formulario lógico se refiere a un tipo de formulario lógico que define el patrón que el formulario lógico debe seguir. El tipo de formulario lógico al que se hace referencia mediante un formulario lógico puede seleccionarse desde múltiples diferentes tipos de formulario lógico para establecer rápidamente el aspecto y contendidos del formulario lógico. En realizaciones de la presente invención, el tipo de formulario lógico son modelos que, cuando se combinan con un negocio u otro modelo de aplicación, resulta en la generación de un formulario lógico.

El tipo de formulario lógico expone el esquema (que describe la estructura del formulario, qué elementos puede contener, etc.) al que el formulario debe ajustarse, los Mapas (o Reglas) que correlacionan (automática o manualmente a través de un desarrollador de aplicación) el modelo de negocio con el modelo lógico y desde este con el modelo físico. Además, el tipo de formulario puede contener código que modifica el comportamiento dinámico del formulario (lógico). Por lo tanto, tipos de formulario exponen información de estilo y disposición, y otros tipos de información específica a un objetivo de visualización. Sin embargo, ya que también especifican reglas para el formulario lógico y sus contenidos tienen un rol mucho mayor. Ciertos aspectos de los tipos de formulario de la presente invención se introducen como se indica a continuación:

Diferentes Tipos de formulario

Como se ha mencionado, en un uso típico de la presente invención, se proporcionan múltiples tipos de formulario diferentes para su uso por un desarrollador de aplicación en la creación de formularios. Por ejemplo, en una realización de ejemplo, tipos de formulario podrían incluir un tipo de formulario de diálogo, un tipo de formulario de Tarjeta o Vista de Tarjeta, un tipo de formulario de Vista de Lista, un tipo de formulario de Vista General de Entidad y un tipo de formulario de Centro de Actividad. Estos tipos de formulario corresponden a diferentes categorías de formularios típicas usadas en una aplicación de negocio en un ejemplo. Por lo tanto, proporcionar los múltiples tipos de formulario permite a desarrolladores de aplicaciones construir todos los formularios que forman una aplicación de negocio de estado de la técnica. Como entenderán los expertos en la materia, estos tipos de formulario particulares son simplemente un ejemplo y la presente invención no se limita a ningún tipo de formulario particular o a ningún número particular de tipos de formulario.

Disposición de tipo de formulario

Los contenidos y estructura de la información de disposición pueden ser diferentes para cada tipo de formulario. Además, la información de disposición puede ser específica del objetivo de visualización. Por ejemplo, la información de disposición para el tipo de formulario de Centro de Actividad podría tener soporte para Temas/Apariencias/Estilos y Páginas Maestras que se usarán en el objetivo de visualización HTML (es decir, red informática mundial o Internet) para la visualización del formulario en una plataforma de sistema operativo particular. Esto proporciona al desarrollador de negocio la libertar para innovar en diferentes objetivos de visualización y retocar la interfaz de usuario de formulario tanto como sea necesario.

Conectable y Extensible

Tipos de formulario ofrecen flexibilidad total y extensibilidad ya que un vendedor de software independiente (ISV) puede modificar o extender los mismos y crear nuevos tipos de formulario, cambiando de este modo el aspecto y comportamiento de toda la aplicación.

Haciendo referencia ahora a las Figuras 3-1 y 3-2, se muestran ilustraciones diagramáticas de un formulario lógico de Orden de Ventas creado usando un tipo de formulario de Tarjeta de acuerdo con una realización de ejemplo de la presente invención. En este ejemplo, el formulario lógico de Orden de Ventas contiene un conjunto de controles lógicos agrupados en tres diferentes grupos de controles lógicos. Como se describirá adicionalmente a continuación, el tipo de formulario de Tarjeta define cómo y dónde aparecerán los controles en los formularios físicos en diferentes objetivos de visualización. Las Figuras 3-1 y 3-2 ilustran la misma información, incluyendo la Figura 3-2 flechas que

ilustran relaciones y orígenes de información entre el tipo de formulario lógico y el formulario lógico, y la generación de un formulario físico a partir del formulario lógico.

Las Figuras 3-1 y 3-2 en forma de diagrama ilustran una instancia 305 de formulario generada usando un tipo 300 de formulario. En un cliente 310 principal, la instancia 305 de formulario es el formulario 306 lógico (mostrado como un modelo conceptual de un formulario) y el tipo 300 de formulario es un tipo 301 de formulario lógico. En un objetivo 320 de visualización, el formulario 305 es un formulario 370 presentado creado usando el formulario 306 lógico, y finalmente a partir del tipo 301 de formulario lógico. El formulario 370 presentado es una implementación particular, pero pueden conseguirse muchas otras implementaciones o presentaciones. El formulario 370 presentado también puede denominarse como un formulario físico.

El tipo 300 de formulario incluye dos partes, un esquema 330 que define qué tiene que incluirse en formularios particulares usando el tipo de formulario, y una disposición 331 que incluye controles que especifican cómo debería presentarse o dibujarse el formulario en un objetivo de visualización específico. Un tipo de formulario también puede incluir una clase detrás de código. Teniendo diferentes disposiciones que pueden usarse por un tipo de formulario, los formularios creados usando el tipo de formulario pueden personalizarse a diferentes objetivos de visualización (por ejemplo, pantallas de teléfonos móviles, pantallas de asistentes digitales personales, monitores de ordenadores personales, etc.). Con el tipo 300 de formulario representando un patrón capturado a usar en múltiples formularios, pueden generarse múltiples instancias 305 de formularios usando el tipo 300 de formulario.

Considérese para este ejemplo un procedimiento a través del cual puede ir un desarrollador para crear un formulario para una entidad de Orden de Ventas o modelo de objeto (es decir, el modelo de negocio o aplicación). En primer lugar, el desarrollador podría mirar para ver entre qué clases de formularios puede elegir. Seleccionar el tipo de formulario "Tarjeta", como se muestra en 325 en las Figuras 3-1 y 3-2, llama o designa el Esquema 330 de Tarjeta como se representa mediante la flecha 326 en la Figura 3-2.

Cuando el desarrollador especifica un tipo de formulario particular y esquema asociado, en algunas realizaciones está eligiendo incluir en el formulario la información o campos impuestos por el esquema, con metadatos del negocio del usuario u otro modelo que rellena los valores de campos. Por ejemplo, seleccionado el tipo 301 de formulario de Tarjeta (y Esquema 330 de Tarjeta asociado), el formulario 306 lógico incluirá un Área 336 de Contenido que corresponde al Área 335 de Contenido definida en el esquema 330. El origen del Área 336 de Contenido que corresponde al Área 335 de Contenido se representa mediante la flecha 337 en la Figura 3-2. Análogamente, porque el esquema 330 del tipo de formulario de Tarjeta incluye un Área 340 de Parte de UI, un Área 345 de Parte de UI de Entidad Relacionada y un Área 350 de Acción, el formulario 306 lógico incluye estas áreas o campos también como se muestra en 341, 346 y 351. De nuevo, los orígenes de estos campos se representan en forma de diagrama mediante las flechas 342, 347 y 352.

Como se ha mencionado anteriormente, cada tipo de formulario también incluye al menos una disposición 331 (normalmente una por objetivo de visualización) que incluye controles que especifican cómo debería presentarse o dibujarse el formulario en un objetivo de visualización específico. Como se muestra en forma de diagrama en las Figuras 3-1 y 3-2, la disposición para el tipo de formulario de Tarjeta incluye controles que provocan que un objetivo 320 de visualización incluya un Área 339 de Contenido, un Área 344 de Parte de UI, un Área 349 de parte de UI de Entidad Relacionada y una Área 354 de Acción. Las flechas 338, 343, 348 y 353 ilustran la correlación de estas áreas en la disposición de tarjeta con sus correspondientes áreas en el esquema de Tarjeta. El formulario 306 lógico, que se genera usando el tipo de formulario lógico seleccionado y metadatos del modelo de aplicación (en este ejemplo una entidad de Orden de Ventas), se presenta como un formulario físico 370 en el objetivo 320 de visualización. Este procedimiento se ilustra en la Figura 3-2 usando las flechas 361 y 362.

MODELOS Y MAPAS

Muchos sistemas de información usan modelos. Ejemplos de modelos son: diagramas de objeto, esquemas de Lenguaje de Marcas Extensible (XML), definiciones de bases de datos y definiciones de formularios. Un modelo a menudo se define como un conjunto de objetos, cada uno de los cuales tiene propiedades, composiciones y asociaciones. En UI de negocios, las jerarquías de control usadas para presentar los formularios pueden considerarse como modelos, tal como árboles de control de Windows y modelos de objetos de Lenguaje de Marcas de Hipertexto (HTML). También, puede usarse sintaxis tal como Lenguaje de Modelización Unificado (UML) para definir un modelo (por ejemplo, las definiciones de clases). En un marco de ejemplo usado para ilustrar los procedimientos de la presente invención, las aplicaciones se modelan usando entidades de negocio. Por lo tanto, el modelo de negocio que consiste en estos objetos de negocio llamados entidades, relaciones entre entidades y propiedades en las entidades. Véase para un ejemplo de un modelo 380 simple las entidades 381, 382, 383 y 384 mostradas en la Figura 4-1. Las entidades tienen propiedades (véasen por ejemplo las propiedades 385 de la entidad 381) y relaciones con otras entidades (véase por ejemplo la relación 386 entre las entidades 381 y 384).

Cuando un modelo se transforma en otro modelo, se usa un mapa explícitamente o en ocasiones implícitamente. Los mapas describen las relaciones entre modelos. Algunos ejemplos incluyen: Transformación de Lenguaje de Hojas de Estilo Extensible (XSLT) que se concibe para correlacionar XML con XML; controles que se usan para presentar un modelo de objeto en una superficie de dispositivo específico; correlaciones de órdenes desde una

aplicación a otra (porque órdenes en diferentes aplicaciones pueden tener formatos diferentes); y herramientas de Ingeniería de Software Asistida por Ordenador (CASE) que correlacionan UML con definiciones de clases.

En aplicaciones de negocio actuales, los mapas se programan en su mayoría usando correlaciones de un objeto a la vez, significando que las correlaciones se codifican como declaraciones de "conmutación" en código, que toman un objeto particular como entrada y devuelven otro objeto. Por lo tanto, aplicaciones de negocio convencionales habitualmente usan mapas imperativos, mapas escritos en el código de un lenguaje de programación típico. Usando modelo a la vez de acuerdo con la presente invención, se presenta que la productividad puede mejorarse por un orden de magnitud. Además de la ganancia de productividad, existe una ganancia mental en la percepción del problema de generación de UI como una correlación de modelos con otros modelos mapas de uso. Además, otro beneficio es el mayor nivel de abstracción encontrado en los mapas definidos declarativamente de la presente invención. La presente invención permite que los mapas sean explícitos y declarativos. En la presente invención, los mapas pueden ser o bien declarativos o bien imperativos (debido al hecho de que uno puede invalidar la correlación en código).

La naturaleza explícita de los mapas significa que los mapas son externos al motor de generación usado para hacer la correlación o presentación, y que los mapas son en sí mismos modelos. Dicho de otra forma, la naturaleza explícita de los mapas significa que se definen de forma separada de los controles y los formularios. Convencionalmente, esta correlación se ha hecho implícitamente dentro del código de controles o código de formularios.
La naturaleza declarativa de los mapas significa que los mapas no son imperativos (codificados en un lenguaje de programación típico). Como se usa en el presente documento, la frase "definido declarativamente" significa que los mapas no solo se definen en código como ha sido el caso convencionalmente, sino que se definen en un formato que permite que los mapas se cambien fácilmente. Ejemplos de un formato definido declarativamente incluyen, pero no se limitan a, documentos XML, ficheros separados por comas, Mapas de BizTalk (correlacionando un esquema de datos con otro) y Mapas de Entidad de MBF (correlacionando un modelo de objeto con un esquema de una base de datos). Pueden usarse una amplia variedad de formatos de correlación declarativos de acuerdo con la presente invención, y qué formato se elige no es de particular importancia. Es importante que el mapa declarativo tenga un conjunto limitado de posibilidades, haciendo por lo tanto más fácil proporcionar una herramienta de diseño intuitiva para definir el mapa. En contraste, un mapa imperativo (usando código) tiene casi ilimitadas posibilidades a través del lenguaje de programación y por lo tanto es extremadamente difícil crear una herramienta de diseño intuitiva. En su lugar, se requieren conocimientos de programación para crear la misma.

Debe observarse que en realizaciones de la presente invención que usan mapas declarativos, los mapas no necesitan ser únicamente declarativos. En instancias en las que es necesario crear un mapa que es demasiado complejo definir declarativamente, pueden incluirse aspectos de correlación imperativa en el de otra manera mapa declarativo. Por ejemplo, pueden crearse funciones complejas e incluirse en el mapa. Un ejemplo podría ser que si una Dirección de Facturación y Dirección de Envío son casi la misma, entonces únicamente se muestra la Dirección de Facturación en el Formulario. El algoritmo de determinación de si dos direcciones son casi la misma podría ser una función definida implícitamente usada en el Mapa.

La presente invención proporciona abstracciones de programación y una arquitectura de perspectiva adecuada para el desarrollo y despliegue de aplicaciones de negocio a base de una arquitectura distribuida y orientada a servicio. El marco aísla la lógica de negocio escrita a estas abstracciones de cambios a tecnologías subyacentes, preservando el activo crítico de un equipo de desarrollo de aplicación de negocio. La presente invención extiende enfoques a desarrollo controlado por modelo, moviéndose de un modelo de tiempo de diseño con generación de código a tener "servicios de aplicación conocedor de modelo" verdaderos, que pueden interpretar el modelo de negocio en tiempo de ejecución.

UI CONTROLADA POR MODELO A BASE DE MAPAS

Tener el modelo de aplicación es una característica importante cuando se genera la UI para una aplicación de negocio construida en realizaciones de la presente invención. Una gran mayoría de la UI puede generarse solamente a base del modelo de la lógica de negocio y mapas. Cuando un desarrollador de aplicación ha modelado una nueva entidad, la UI se obtiene a partir de esta. Esto se ilustra en forma de diagrama en la Figura 4-2 que ilustra el modelo de negocio 380 que se correlaciona (como se muestra en 388) con un modelo de UI 390. La flecha 388 representa el procedimiento de correlación, así como un motor de correlación configurado adecuadamente que usa un mapa para llevar a cabo el procedimiento de correlación.

Aunque esta correlación puede conseguirse usando técnicas de codificación tradicionales, la correlación no es tan sencilla si deben cumplirse ciertos desafíos. El desafío es que cuando se crean y usan nuevos tipos de propiedades en una entidad, la transformación codificada podría no saber cómo tratar el nuevo tipo y la transformación por lo tanto tiene que modificarse y recompilarse. Otro desafío es tratar controles desarrollados recientemente que únicamente serán de valor si se incluyen en la transformación - de nuevo esto resulta en la preprogramación de la transformación. Las técnicas de correlación de la presente invención son capaces de cumplir estos desafíos. Obsérvese que cualquier modificación de la UI en tiempo de ejecución (a través de código) también puede considerarse una correlación. La plataforma usada en la presente invención expone un modelo de UI por capas y

usa mapas para transformar modelos desde una capa a otra. Esto se describe a continuación en mayor detalle.

Los procedimientos y aparato de la presente invención proporcionan una forma de calcular cómo presentar información de negocio al usuario en una plataforma dada. La invención se construye sobre la correlación de modelos con otros modelos, trabajando desde un modelo abstracto (describiendo las entidades de negocio con las que interactuar) hasta un modelo concreto (especificando exactamente qué control específico del dispositivo debería usarse para presentar la información de negocio). En general, esta correlación puede implicar cualquier número de etapas.

Por ejemplo, considérese el diagrama 400 de bloques mostrado en la Figura 5-1 que ilustra un proceso de correlación desde un modelo 405 maestro a un modelo 425 especializado usando dos etapas de correlación explícitas y declarativas. El modelo 405 maestro (es decir, "modelo A") puede ser, por ejemplo, una base de datos, tabla, entidad, objeto u otros tipos de modelos en un dominio de problema específico a un usuario. El modelo 405 maestro se correlaciona con un modelo 415 intermedio (es decir, "modelo B") con la etapa de correlación ilustrada en 411 usando un mapa 410 (es decir, "mapa A-B"). El modelo 415 intermedio puede ser un modelo independiente del objetivo de visualización que tiene controles lógicos, como se describirán a continuación en mayor detalle. El modelo 415 intermedio se correlaciona a continuación con un modelo 425 especializado (es decir, "modelo C") con la etapa de correlación ilustrada en 421 usando un segundo mapa 420 (es decir, "mapa B-C"). El modelo 425 especializado puede ser un modelo específico de objetivo de visualización que tiene controles físicos, como también se describirá a continuación en mayor detalle. Las flechas usadas para representar las etapas 411 y 421 de correlación también representan motores de correlación que se configuran para utilizar los mapas 410 y 420 para implementar las etapas de correlación.

De acuerdo con algunas realizaciones de la presente invención, el esquema de correlación implicado en la determinación de cómo permitir que el usuario interactúe con información de negocio en la plataforma cliente implica al menos tres etapas, como se describe a continuación y como se muestra en forma de diagrama en el diagrama 450 de bloques de la Figura 5-2. El modelo 455 inicial (véase también el modelo 405 maestro mostrado en la Figura 5-1) contiene información acerca de las entidades de negocio con las que el usuario debe interactuar. Cada dato de este modelo de aplicación es de un tipo particular. La primera etapa implica la determinación de qué control lógico emplear para un tipo dado (cadena, número entero, tipo decimal que representa valores monetarios, direcciones que contienen otros valores, etc.) del dato a presentar.

El control lógico a usar para el tipo dado se determina usando una correlación desde tipo de datos en el modelo 455 con control lógico en el modelo 465. El procedimiento de correlación se ilustra en 461, y utiliza un mapa 460 (es decir, el "mapa de tipo de dato a control lógico"). Controles lógicos tienen varias propiedades útiles. No tienen ninguna dependencia de ningún objetivo de visualización específico, pero mantienen propiedades que gobiernan el comportamiento de controles físicos específicos de dispositivo. La consulta del control lógico se realiza teniendo en cuenta el tipo jerarquía. Si ningún control lógico es específicamente adecuado para el encapsulado de las propiedades de un tipo específico, la búsqueda continúa con un tipo base, hasta que se encuentra un control lógico para tratar el tipo.

Una vez que se ha identificado un control lógico a partir del tipo de datos a representar, debe encontrarse el control físico usado para realizar realmente la presentación en la plataforma dada. Estos controles físicos se denominan en ocasiones como "adaptadores". Esto se hace usando otra correlación, produciendo el control físico del control lógico y el objetivo de visualización. El procedimiento de correlación se ilustra en 471 y usa el mapa 470 (es decir, el "mapa de control de control lógico a control físico ") para generar el modelo 475 de control físico a partir del modelo 465 de control lógico.

Cuando el cliente se ejecuta en el objetivo de visualización del usuario, el control físico se usará para crear instancias de los controles nativos usados para interactuar con el usuario. Esto se hace mediante una tercera correlación, produciendo un conjunto de controles nativos a partir del control físico. Por ejemplo, si el control físico fuera un control de dirección, el control físico correlacionaría con controles nativos para calle, ciudad, país. El procedimiento de correlación se ilustra en 481 y usa el mapa 480 (es decir, el "mapa de control físico a control nativo") para generar el modelo 485 de control nativo a partir del modelo 475 de control físico. En algunas realizaciones, esto es un mapa imperativo - pero no tiene por qué serlo. De nuevo, las flechas 461, 471 y 481 también representan el motor o motores de correlación usados para implementar las funciones de correlación como se especifica mediante los mapas 460, 470 y 480.

La correlación descrita anteriormente puede aumentarse con otras correlaciones para conseguir el resultado deseado. Otros factores incluyen el tipo de formulario presentado (tarjeta o vista de lista), el rol de usuario (posiblemente limitando la información ofrecida al usuario). El procedimiento de llegada desde el modelo abstracto al modelo concreto es puramente prescriptivo (describiendo las correlaciones implicadas) y se permite flexibilidad siendo capaz de cambiar estas correlaciones.

Como otro ejemplo, la Figura 5-3 ilustra un diagrama 500 de bloques que muestra un procedimiento de correlación para conseguir desde un nombre del usuario y número de identificación (ID) hasta el HTML usado para presentar esta información en un navegador. El modelo 505 de negocio maestro inicial es una entidad (u objeto) o clase de

entidades (o clase de objetos) que tiene el nombre del cliente e ID como propiedades. Las propiedades de "Nombre" e "ID" del modelo 505 son de tipos "Cadena" y "Número", respectivamente. El modelo 505 se correlaciona con una capa de control lógico del modelo 515 usando un mapa 510 prescriptivo. El procedimiento de correlación se representa en 511. En este ejemplo, los datos tipo "Cadena" se correlacionan con un control lógico de "Caja de Texto", mientras los datos tipo "Número" se correlaciona con un control lógico de "Caja de Números".

A continuación, el modelo 515 de control lógico se correlaciona con un modelo 525 HTML usando el mapa 520. El procedimiento de correlación se representa en 521. En este ejemplo, el modelo 525 es un modelo de control físico en forma de un modelo de HTML. Por lo tanto, el mapa 520 correlaciona los controles lógicos del modelo 515 con etiquetas de HTML o elementos en el modelo 525. El modelo 525 de HTML se usa continuación para presentar la información desde el modelo 505 en un navegador. De nuevo, las flechas usadas para representar las etapas 511 y 521 de correlación también representan motores de correlación configurados adecuadamente que utilizan los mapas 510 y 520 para implementar el procedimiento de correlación.

La Figura 5-4 ilustra un aspecto adicional de realizaciones de la presente invención en el que pueden relacionarse varios tipos de propiedades diferentes con los mismos controles finales, así que el número de controles requeridos no aumenta necesariamente cuando el número de tipos de propiedades aumenta. Como se muestra en el diagrama 550 de bloques de la Figura 5, un modelo 560 de negocio que tiene propiedades 561 de diferentes tipos se correlaciona con un modelo 580 de objetivo de visualización usando los mapas 555. Similar a ejemplos anteriormente analizados, el modelo 560 se correlaciona con un modelo 570 de capa lógica que tiene controles 571 lógicos. El motor de correlación y procedimiento de correlación, que usan el mapa 565, se ilustran en 566. El mapa 565 correlaciona los tipos de dato ("Tipo de ID", "Cadena" y "Flotante") de las propiedades 561 del modelo 560 con controles lógicos ("Número" y "Cadena"). En este caso, tanto los tipos de dato "Tipo de ID" como "Flotante" se correlacionan con el tipo de control lógico de "Número", mientras el tipo de dato de "Cadena" se correlaciona con el tipo de control lógico de "Cadena".

A continuación, modelo 570 de capa lógica se correlaciona con modelo 580 de objetivo de visualización que tiene controles 581 físicos específicos para un objetivo de visualización particular. El modelo 570 se correlaciona con el modelo 580 usando el mapa 575, con el procedimiento y motor de correlación representado en 576. El mapa 575 correlaciona los tipos de control lógico "Número" y "Cadena" del modelo 570 con el tipo de control físico "Caja de Texto" del modelo 580, ilustrando de nuevo que varios tipos diferentes de un modelo particular pueden relacionarse con un único tipo en otro modelo. Por extensión, varios tipos diferentes de propiedades de un modelo de negocio pueden relacionarse con el mismo control final (por ejemplo "físico").

Experiencia de Desarrollador

Cuando un desarrollador crea una nueva entidad, que se construye únicamente a partir de tipos existentes, también se construye una UI por defecto a través de los mapas. Si la UI por defecto no ofrece la experiencia de usuario deseada, el desarrollador puede elegir:

- Modificar el modelo de negocio para reflejar los requisitos. Por ejemplo, en algunas realizaciones si la ordenación de una secuencia de propiedades está mal, digamos que el ID debería visualizarse antes del Nombre, entonces la entidad podría editarse. En otras realizaciones, la ordenación de propiedades reside en un "comportamiento" externo. Sin embargo, una agrupación ordenada de propiedades puede "añadirse" a la entidad - y esto puede usarse para hacerlo. Por lo tanto, en estas realizaciones, el mejor enfoque podría ser cambiar el orden en el agrupamiento.
- Modificar el modelo de formulario lógico generado. Conmutar el Nombre y ID también podría hacerse en el formulario. Esto potencialmente presenta algunos desafíos de mantenimiento posteriormente si se cambia la lógica de negocio, tal como hace el cambio en la lógica de negocio que sobrescribe cambios en el formulario. También, puede ser necesario cambiar la lógica de negocio en cada formulario.
- Modificar el mapa. Si el ID se correlaciona con un control de Número pero es más adecuado un control de Cadena, el mapa es el lugar correcto para hacer el cambio.

Existe un número de beneficios al cambio del mapa en lugar de la más tradicional modificación de modelos. En primer lugar, los cambios pueden tener un ámbito más amplio. Si se cambiase la entrada de mapa usada en el ejemplo anterior, todas las entidades que usan el "Tipo de ID" automáticamente obtendrían la actualización. Esto resultaría en una UI muy consistente, de la que podría beneficiarse el usuario final.

Otro beneficio se vuelve evidente cuando se mira al mantenimiento y versiones futuras de una aplicación. Cambiando la forma en que se generan modelos, pero sin cambiar los modelos generados, el modelo maestro puede actualizarse, y en lo sucesivo los modelos dependientes pueden generarse sin ningún riesgo de conflicto. No regenerar en absoluto los formularios podría resultar en inconsistencias entre entidades y los formularios usados para ver y editar las mismas. Los mapas también descomponen la gran tarea de generación en varias entradas de mapa declarativas pequeñas.

Si el desarrollador crea un nuevo tipo de propiedad, tal como "Dinero", esto puede usarse de forma instantánea porque la UI se generará eficientemente si únicamente se añade una única entrada de mapa. En este ejemplo la

nueva propiedad "Dinero" podría correlacionarse a un control de "Número". El desarrollador podría también elegir explotar la información de metadatos añadida, y crea un control de "Dinero", y tiene la propiedad correlacionada al nuevo control. La tecnología de correlación hace que ambos escenarios sean válidos.

El Lenguaje de Correlación

La correlación usa un lenguaje de correlación declarativo simple que es extensible. La correlación toma uno o más testigos como entrada y devuelve uno o más testigos como salida. Dado un tipo de propiedad como entrada, uno o más controles lógicos pueden especificarse como salida. También debe ser posible especificar la salida como nula. Por ejemplo, el "Tipo de ID" podría ser un campo generado por ordenador, que el usuario no puede editar o ver, en cuyo caso el tipo se correlaciona a nada. También, la correlación puede controlar parámetros en la salida. Por ejemplo, la propiedad "Cadena" podría resultar en una Caja de Texto más ancha en comparación con las otras Cajas de Texto en el formulario.

Para tratar el problema de alcance abordado antes, se necesita una condición de ámbito - en el formulario de ejemplo anteriormente analizado, "Tipos de ID" se correlacionan con "Controles de ID", pero en todos los demás formularios se usa un control "Número". También pueden usarse otros parámetros como ámbito, incluyendo la entidad de negocio, el estereotipo de entidad, el tipo de formulario, etc. Existen otras condiciones que serían beneficiosas tener en consideración cuando debe realizarse una correlación. Un ejemplo es el control padre. Si el control padre es una lista, una propiedad de enumerador podría elegir correlacionarse a una lista desplegable y no un botón de radio. Otra condición podría ser el número de posibles selecciones en el enumerador; podrían usarse botones de radio si hay dos o tres, pero más elecciones podrían resultar en una lista. Siguiendo este camino, el lenguaje de correlación terminará siendo muy complejo en comparación al requisito inicial, y se necesita otro nivel de abstracción encima del mapa para que los desarrolladores entiendan los mapas. Este patrón se ha visto con Transformación de Lenguaje de Hojas de Estilo Extensibles (XSLT), en la que varias herramientas se han implementado para ocultar la complejidad.

FORMULARIOS LÓGICOS - UN MODELO DE UI

Cuando se correlaciona desde el modelo de la lógica de negocio al modelo de UI, se inserta una capa independiente de la disposición, también llamada una capa lógica. Si se cree que el modelo de la lógica de negocio puede relacionarse con la UI final independientemente del objetivo de visualización, la capa lógica es una abstracción sencilla. Algunos metadatos serán comunes para todos los objetivos de visualización tal como la propia entidad de negocio, y algunas partes serán específicas para el objetivo de visualización específico. La capa lógica es la parte común.

La Figura 6 es una ilustración diagramática de las actividades de tiempo de diseño y actividades de tiempo de ejecución usadas para crear formularios. En tiempo de diseño, se usan las herramientas 605 de modelado para crear modelos o definiciones de formulario y mapas tal como los analizados anteriormente. Estas definiciones de formulario y mapas pueden almacenarse en una base 610 de datos de metadatos.

En tiempo de ejecución, los modelos o formularios se correlacionan con el modelo 625 de capa lógica. El modelo 625 de capa lógica también se genera usando datos de tiempo de ejecución almacenados en la base 615 de datos aplicados a la lógica 620 de negocio. También en tiempo de ejecución, el modelo 625 de capa lógica se correlaciona con un modelo 630 de objetivo de visualización como se describe anteriormente.

Los formularios que incluyen capas lógicas y controles es el puente entre la lógica 620 de negocio y los objetivos 630 de visualización. Tiene conocimiento limitado de disposición y conocimiento limitado de la lógica de negocio. La capa lógica define el contenido de un formulario a base de las entidades de negocio y trata cuestiones de tiempo de ejecución comunes tal como formularios de vinculación de datos a la instancia de tiempo de ejecución de las entidades de negocio. Adicionalmente, la capa lógica trata la seguridad común a todos los objetivos de visualización; proporciona metadatos a cada objetivo de visualización y la capa lógica puede tratar validación de entradas.

Un arquitecto de negocio o desarrollador puede centrarse en lógica de negocio específica de dominio y datos. Cuando el foco se cambia a la UI, los detalles de disposición, cuestiones de vinculación de datos, código de establecimiento, validación de entrada, ocultación de propiedades no legibles, tratamiento de errores, etc., se ocultan todos en el nivel superior de abstracción encontrado en la capa lógica. El especialista de dominio puede centrarse en los contenidos de la UI - qué tiene sentido que vea el usuario - y no necesita tener un conocimiento profundo acerca de objetivos de visualización específicos y sus diferentes tecnologías de presentación.

Como se ha analizado, los formularios lógicos o modelos de capas lógicas se construyen usando controles lógicos. Pueden añadirse fácilmente nuevos controles, haciendo la capa lógica muy flexible y extensible. Cuando se desarrolla un nuevo control, simplemente se añade a formularios existentes cambiando los mapas usados. Cada objetivo de visualización se beneficiará de la nueva funcionalidad sin tener que implementar nuevos controles, pero si tiene sentido, puede introducirse un nuevo control.

OBJETIVOS DE VISUALIZACIÓN

Los formularios lógicos y controles se correlacionan con tecnologías de presentación específicas usadas por los objetivos de visualización. Como en otras Figuras, esto se ilustra en la Figura 7 en la que el modelo de capa lógica o formulario 705 se correlaciona con varios objetivos de visualización específicos. En este ejemplo particular, el objetivo 710 de visualización usa tecnología de presentación de Windows, mientras el objetivo 715 de visualización usa una tecnología de presentación Web. Los objetivos de visualización son responsables del tratamiento de todas las interacciones de usuario, incluyendo la presentación de los formularios y controles y tratamiento de la entrada de usuario. Cada objetivo de visualización necesita un número de controles de modo que los controles en la capa lógica se correlacionan con algo válido. Es decir, la propiedad tiene que ser compatible con los tipos de valor que puede tratar el control y el control debería presentar ese valor de una forma razonable. En otras palabras, no existe un número específico de controles que necesiten estar disponibles en cada objetivo de visualización, ya que la tecnología de correlación tiene un significativo impacto en esto.

Los objetivos de visualización controlan la interacción de usuario y esencialmente también el paradigma de interacción. Una página web y una ventana de Formularios de Windows podrían generarse a base del mismo formulario lógico, pero si usan una política de interacción de comunicación o una política de envío posterior sólida se determina naturalmente mediante el objetivo de visualización. Cada objetivo de visualización elige cuánto se muestra de un formulario al usuario. Un formulario de Windows puede ocultar información en páginas de pestañas, mientras una página web puede elegir mostrar toda la información a la vez. Estas decisiones se toman a base del formulario lógico y por lo tanto también el tipo de formulario, que obtienen los objetivos de visualización. Diferentes objetivos de visualización necesitan información adicional para tomar tales decisiones de paginación y de forma similar los formularios lógicos y controles pueden anotarse con información de objetivo de visualización específica.

COMPORTAMIENTOS - LÓGICA DE FORMULARIO DECLARATIVA

Ya que los formularios se hacen de controles lógicos, que definen el contenido, existe aún una necesidad de añadir dinámicas a los formularios. Esto podría hacerse usando código. Pero no es fácil correlacionar código en un formulario y por lo tanto se necesita un nivel de abstracción declarativo. Al mismo tiempo, existen muchos patrones en el código, que se añaden a los formularios. Por ejemplo, en muchos formularios se encuentra código para deshabilitar un campo si se rellena otro campo. Si se selecciona "En efectivo" como un tipo de pago en un formulario, el "Número de Tarjeta de Crédito" se sombrea o incluso se oculta.

Estos patrones se capturan en un concepto llamado "comportamientos." Los comportamientos se añaden declarativamente a los formularios y se activan a través de eventos en el formulario. Los comportamientos pueden, dependiendo de valores y propiedades de los controles, establecer propiedades en otros controles, pero los comportamientos no se limitan a este uso. Los comportamientos se correlacionan principalmente a base de limitaciones en las entidades o en otras propiedades de metadatos. Las entidades son responsables del tratamiento de la lógica de negocio y los comportamientos son responsables de la lógica de UI. En otras palabras, los comportamientos no deberían implementar ninguna lógica de negocio ya que esto violaría la separación de asuntos.

SISTEMA POR CAPAS Y EJEMPLO DE PROCEDIMIENTO

Haciendo referencia ahora a la Figura 8, se muestra una ilustración diagramática de un sistema por capas y procedimiento que implementa los diversos conceptos descritos anteriormente (es decir, tipos de formulario lógico, mapas de UI, controles, comportamientos, etc.) en combinación. Como se muestra en la Figura 8, el sistema o procedimiento 900 se implementa usando un marco 905 de negocio que puede incluir aplicaciones de negocio y sistemas informáticos, un motor de correlación de cliente principal o componente 910 y uno o más objetivos 915 de visualización. Para fines de ilustración, la Figura 8 se segmenta para mostrar los mapas 920 (incluyendo mapas 921 y 922), modelos 925 (incluyendo aplicación o modelo 926 de negocio, modelo 927 de formulario lógico y modelo 928 físico) y librerías 930.

Las librerías 930 incluyen librerías 931 de tipos de propiedades, en el marco 905 de negocio, que define los diversos tipos 923 de propiedades de dato en el modelo 926 de aplicación. En el cliente 910 principal, las librerías 930 incluyen la librería 932 de controles lógicos, librería 933 de comportamientos y librería 934 de tipos de formulario lógico. Como se ha descrito anteriormente, la librería 932 de controles lógicos define los diversos controles 924 lógicos con los que pueden relacionarse el dato o tipos 923 de propiedades del modelo 926. Esta correlación entre el modelo 926 de aplicación y modelo 927 de formulario lógico se realiza mediante el motor de correlación de cliente principal usando el mapa 921.

La librería 933 de comportamientos y librería 934 de tipos de formulario lógico se usan por el motor 910 de correlación de cliente principal en el procedimiento de diseño y generación de formularios como se ha descrito anteriormente. Las librerías 930 también incluyen librería 935 de estilos, librería 936 de controles físicos o nativos y librería 937 de tipos de formulario que se usan por el objetivo 900 de visualización para generar el modelo 928 de formulario físico y por lo tanto para presentar el formulario físico. La librería 936 de control define los diversos controles físicos 925 para un objetivo de visualización particular a los que pueden relacionarse los controles 924 lógicos usando el mapa 922.

REIVINDICACIONES

1. Un procedimiento implementado por ordenador de generación de una interfaz de usuario de formulario controlada por modelo para representar un modelo de aplicación, comprendiendo el procedimiento:

recibir una entrada de selección para seleccionar cuál de una pluralidad de diferentes tipos de formulario lógico usar para generar una interfaz de usuario de formulario para representar un modelo de aplicación;
proporcionar un primer mapa declarativo;
generar un formulario lógico independiente del objetivo de visualización usando el modelo de aplicación, el tipo de formulario lógico seleccionado y el primer mapa declarativo, en el que generar el formulario lógico independiente del objetivo de visualización usando el primer mapa declarativo comprende además correlacionar tipos de propiedades de dato del modelo de aplicación con controles lógicos independientes del objetivo visualización en el formulario lógico independiente del objetivo de visualización;
correlacionar el formulario lógico independiente del objetivo de visualización con un formulario físico usando un segundo mapa declarativo, en el que el formulario físico tiene una pluralidad de controles físicos disponibles para su uso en la presentación del formulario lógico en un objetivo de visualización, comprendiendo el uso del segundo mapa declarativo correlacionar cada uno de los controles lógicos en el formulario lógico con uno de la pluralidad de controles físicos disponibles; y
presentar la interfaz de usuario de formulario en tiempo de ejecución usando el formulario lógico generado de tal forma que el modelo de aplicación está accionado en un tiempo de ejecución, comprendiendo la presentación del formulario lógico como el formulario físico en el objetivo de visualización, en el que el formulario lógico se genera en tiempo de ejecución usando tanto el tipo de formulario lógico seleccionado como metadatos del modelo de aplicación.

2. El procedimiento de la reivindicación 1, en el que generar el formulario lógico independiente del objetivo de visualización comprende generar un modelo de formulario lógico independiente del objetivo de visualización.

3. El procedimiento de la reivindicación 1, en el que generar el formulario lógico independiente del objetivo de visualización usando el modelo de aplicación, el tipo de formulario seleccionado y el primer mapa declarativo comprende además también adjuntar comportamientos aplicados declarativamente al formulario lógico independiente del objetivo de visualización.

4. El procedimiento de la reivindicación 3, en el que los comportamientos aplicados declarativamente se activan mediante eventos en el formulario.

5. El procedimiento de la reivindicación 4, en el que los comportamientos aplicados declarativamente son patrones de lógica que, dependiendo de valores y propiedades de los controles lógicos, establece propiedades en otros controles.

6. El procedimiento de la reivindicación 1, en el que el primer mapa declarativo es externo a un motor de correlación usado para generar el formulario lógico independiente del objetivo de visualización.

7. El procedimiento de la reivindicación 6, en el que el primer mapa declarativo es un modelo usado por el motor de correlación para generar el modelo de control lógico.

8. El procedimiento de la reivindicación 1, en el que cada uno de la pluralidad de diferentes tipos de formulario lógico tiene un esquema asociado que define datos de modelo de aplicación a incluir en el formulario lógico generado, y en el que generar el formulario lógico en tiempo de ejecución comprende además generar el formulario lógico usando el esquema asociado.

9. El procedimiento de la reivindicación 8, en el que el esquema de cada uno de la pluralidad de diferentes tipos de formulario lógico representa patrones capturados de una pluralidad de formularios.

10. El procedimiento de la reivindicación 8, en el que cada uno de la pluralidad de tipos de formulario lógico tiene al menos una disposición definida, comprendiendo el procedimiento además generar el formulario físico en tiempo de ejecución usando la al menos una disposición definida.

11. Un medio legible por ordenador que tiene instrucciones ejecutables por ordenador de realización de etapas de generación de interfaz de usuario de formulario que comprenden:

recibir una entrada de selección para seleccionar cuál de una pluralidad de diferentes tipos de formulario lógico usar para generar una interfaz de usuario de formulario para representar un modelo de aplicación;
proporcionar un primer mapa declarativo;
generar un formulario lógico independiente del objetivo de visualización usando el modelo de aplicación, el tipo de formulario lógico seleccionado y el primer mapa declarativo que comprende correlacionar tipos de propiedades de dato del modelo de aplicación con controles lógicos independientes del objetivo visualización en el formulario lógico independiente del objetivo de visualización;
correlacionar el formulario lógico independiente del objetivo de visualización con un formulario físico usando un

segundo mapa declarativo, en el que el formulario físico tiene una pluralidad de controles físicos disponibles para su uso en la presentación del formulario lógico en un objetivo de visualización, y en el que presentar el formulario lógico al formulario físico usando el segundo mapa declarativo comprende además usar el segundo mapa declarativo para correlacionar cada uno de los controles lógicos en el formulario lógico con uno de la pluralidad de controles físicos disponibles; y

presentar la interfaz de usuario de formulario en tiempo de ejecución usando el formulario lógico generado de tal forma que el modelo de aplicación está accionado en un tiempo de ejecución, comprendiendo la presentación del formulario lógico como el formulario físico en el objetivo de visualización, en el que el formulario lógico se genera en tiempo de ejecución usando tanto el tipo de formulario lógico seleccionado como metadatos del modelo de aplicación.

12. El medio legible por ordenador de la reivindicación 11, en el que generar el formulario lógico independiente del objetivo de visualización usando el modelo de aplicación, el tipo de formulario seleccionado y el primer mapa declarativo comprende además adjuntar comportamientos aplicados declarativamente al formulario lógico independiente del objetivo de visualización.

13. El medio legible por ordenador de la reivindicación 11, en el que el primer mapa declarativo es externo a un motor de correlación usado para generar el formulario lógico independiente del objetivo de visualización.

14. El medio legible por ordenador de la reivindicación 11, en el que cada uno de la pluralidad de diferentes tipos de formulario lógico tiene un esquema asociado que define datos de modelo de aplicación a incluir en el formulario lógico generado, y en el que generar el formulario lógico comprende además generar el formulario lógico usando el esquema asociado.

15. El medio legible por ordenador de la reivindicación 14, en el que el esquema de cada uno de la pluralidad de diferentes tipos de formulario lógico representa patrones capturados de una pluralidad de formularios.

16. El medio legible por ordenador de la reivindicación 14, en el que cada uno de la pluralidad de tipos de formulario lógico tiene al menos una disposición definida, teniendo el medio legible por ordenador adicionalmente instrucciones ejecutables por ordenador de realización de la etapa adicional de generación del formulario físico en tiempo de ejecución usando la al menos una disposición definida.

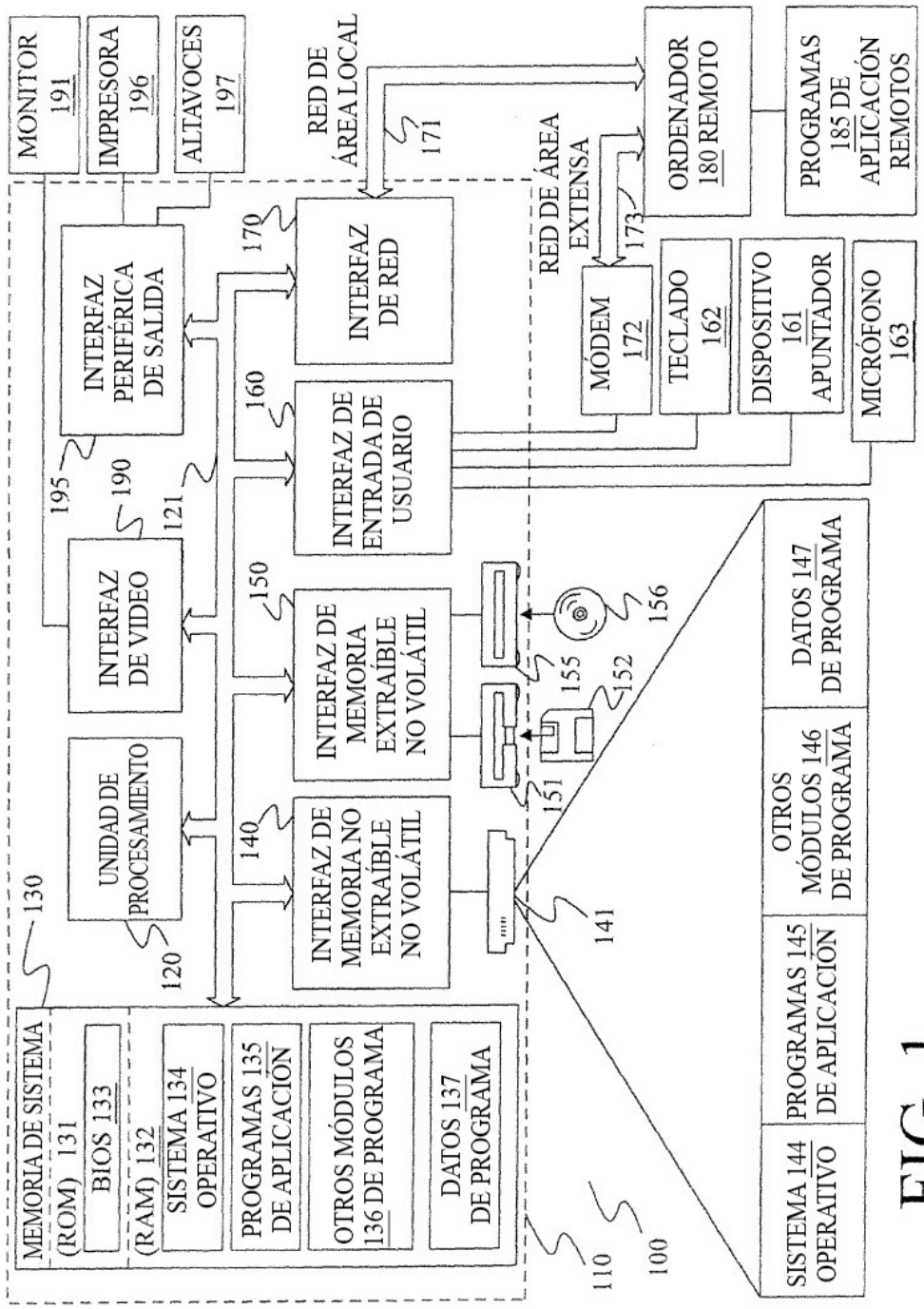
MONITOR 191
IMPRESORA 196
ALTAVOCES 197
RED DE ÁREA LOCAL
171
RED DE ÁREA EXTENSA
ORDENADOR 180 REMOTO
PROGRAMAS 185 DE APLICACIÓN REMOTOS
INTERFAZ PERIFÉRICA DE SALIDA
INTERFAZ DE RED
INTERFAZ DE ENTRADA DE USUARIO
MÓDEM 172
TECLADO 162
DISPOSITIVO 161 APUNTADOR
MICRÓFONO 163
INTERFAZ DE VIDEO
UNIDAD DE PROCESAMIENTO
INTERFAZ DE MEMORIA EXTRAÍBLE NO VOLÁTIL
INTERFAZ DE MEMORIA NO EXTRAÍBLE NO VOLÁTIL
MEMORIA DE SISTEMA
(ROM) 131
BIOS 133
(RAM) 132
SISTEMA 134 OPERATIVO
PROGRAMAS 135 DE APLICACIÓN
OTROS MÓDULOS 136 DE PROGRAMA
DATOS 137 DE PROGRAMA
SISTEMA 144 OPERATIVO
PROGRAMAS 145 DE APLICACIÓN
OTROS MÓDULOS 146 DE PROGRAMA
DATOS 147 DE PROGRAMA
100
110
120
121
130
140
141
150
151
152
155
156
160
170
173
190
195

FIG. 1

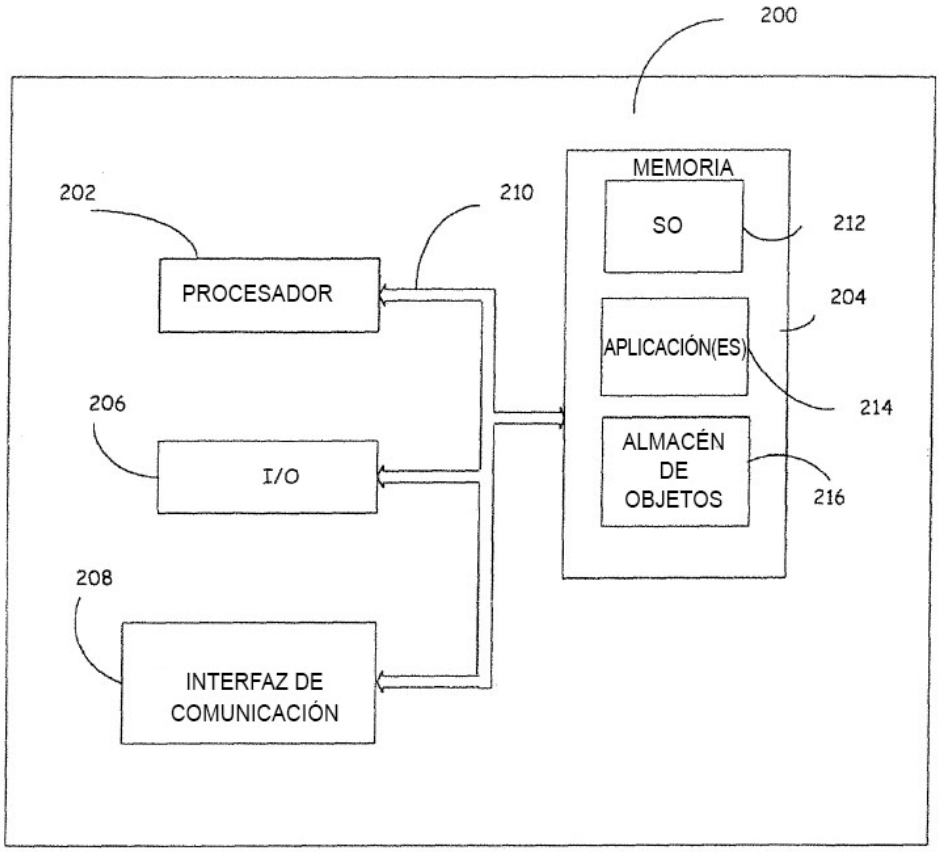
200
MEMORIA
202
210
SO
212
PROCESADOR
204
APLICACIÓN(ES)
214
206
I/O
ALMACÉN DE OBJETOS
216
208
INTERFAZ DE COMUNICACIÓN

FIG. 2

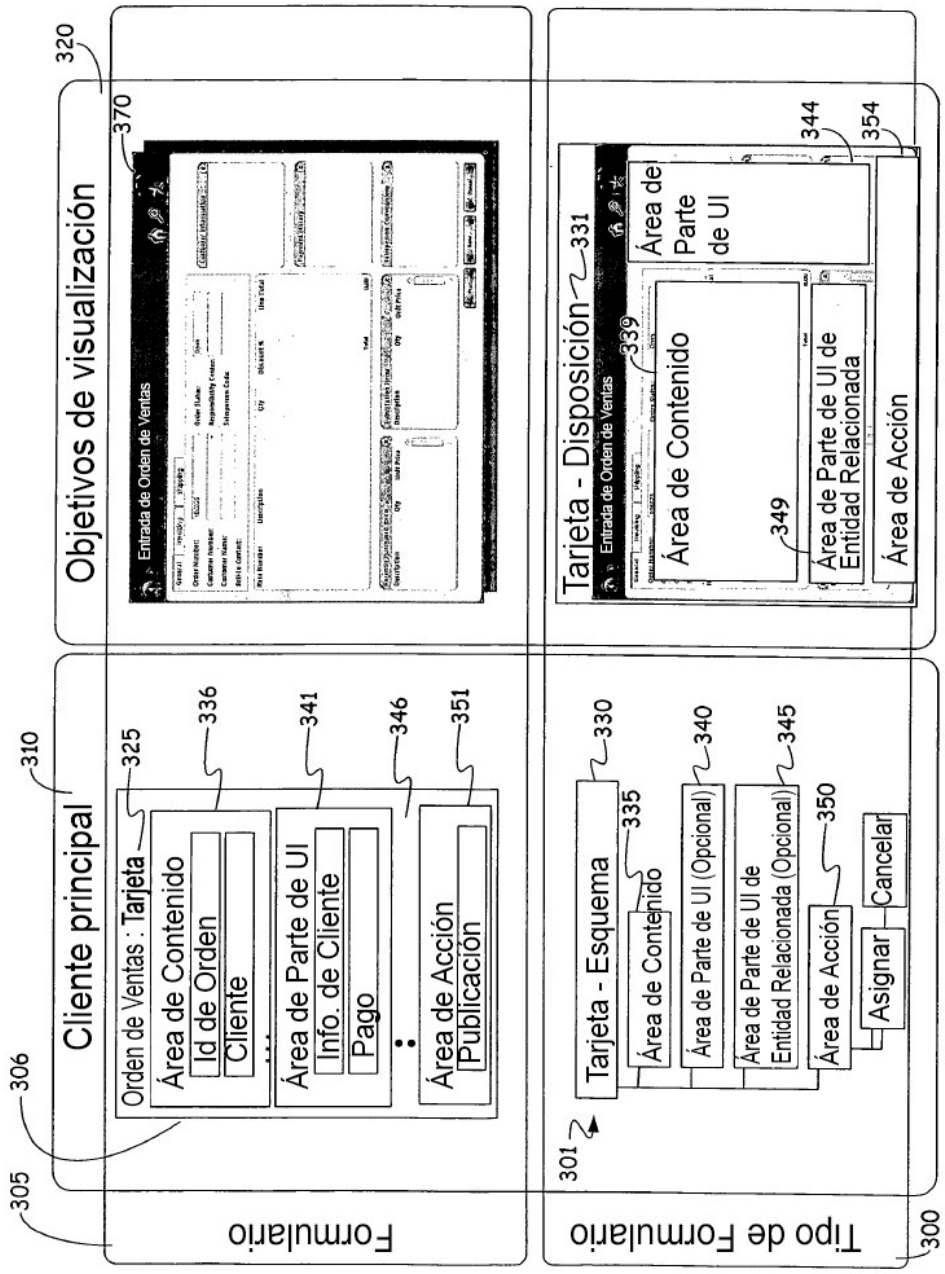
Formulario
Tipo de Formulario
Cliente principal
Orden de Ventas : Tarjeta
Área de Contenido
Id de Orden
Cliente
Área de Parte de UI
Info. de Cliente
Pago
Área de Acción
Publicación
Tarjeta - Esquema
Área de Contenido
Área de Parte de UI (Opcional)
Área de Parte de UI de Entidad Relacionada (Opcional)
Área de Acción
Asignar
Cancelar
Objetivos de visualización
Entrada de Orden de Ventas
Tarjeta - Disposición
Área de Contenido
Área de Parte de UI
Área de Parte de UI de Entidad Relacionada
Área de Acción
300
301
305
306
310
320
325
330
335
336
339
340
341
344
345
346
349
350
351
354
370

FIG. 3-1

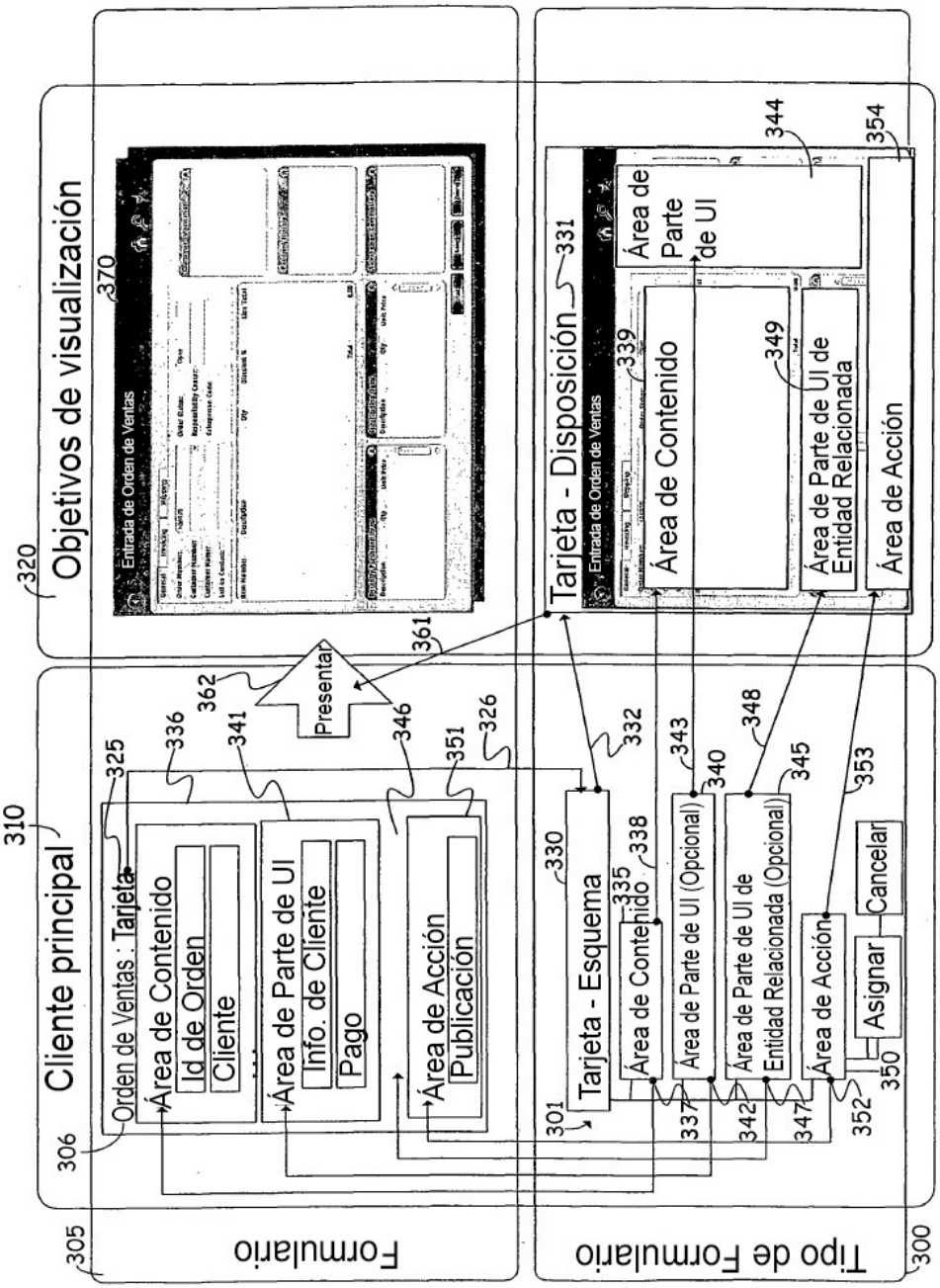
Objetivos de visualización
320
370
Entrada de Orden de Ventas
Cliente principal
310
Orden de Ventas : Tarjeta
306
Área de Contenido
Id de Orden
Cliente
Área de Parte de UI
Info. de Cliente
Pago
Área de Acción
Publicación
325
336
341
346
351
326
Formulario
305
Presentar
362
361
Tarjeta - Disposición
331
Área de Parte de UI
344
Área de Contenido
339
349
Área de Parte de UI de Entidad Relacionada
Área de Acción
354
Tarjeta - Esquema
330
301
Área de Contenido
335
338
Área de Parte de UI (Opcional)
340
343
Área de Parte de UI de Entidad Relacionada (Opcional)
345
348
Área de Acción
353
Asignar
Cancelar
350
337
342
347
352
332
Tipo de Formulario
300

FIG. 3-2

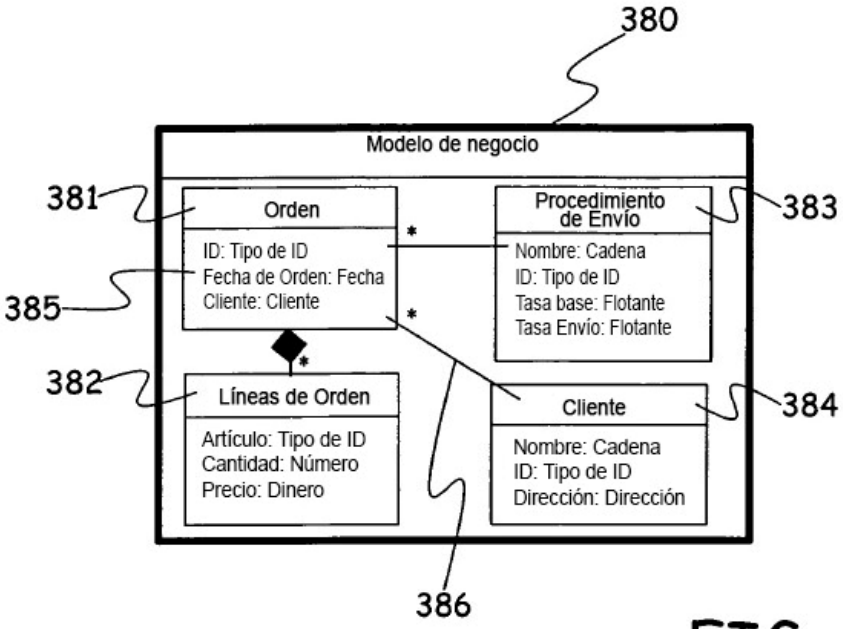
380
Modelo de negocio
381
Orden
ID: Tipo de ID
Fecha de Orden: Fecha
Cliente: Cliente
383
Procedimiento de Envío
Nombre: Cadena
ID: Tipo de ID
Tasa base: Flotante
Tasa Envío: Flotante
385
382
Líneas de Orden
Artículo: Tipo de ID
Cantidad: Número
Precio: Dinero
384
Cliente
Nombre: Cadena
ID: Tipo de ID
Dirección: Dirección
386

FIG. 4-1

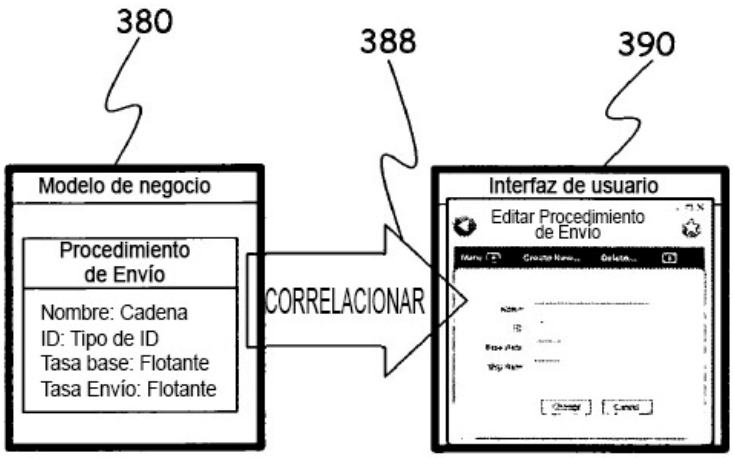
380
388
390
Modelo de negocio
Procedimiento de Envío
Nombre: Cadena
ID: Tipo de ID
Tasa base: Flotante
Tasa Envío: Flotante
CORRELACIONAR
Interfaz de usuario
Editar Procedimiento de Envío

FIG. 4-2

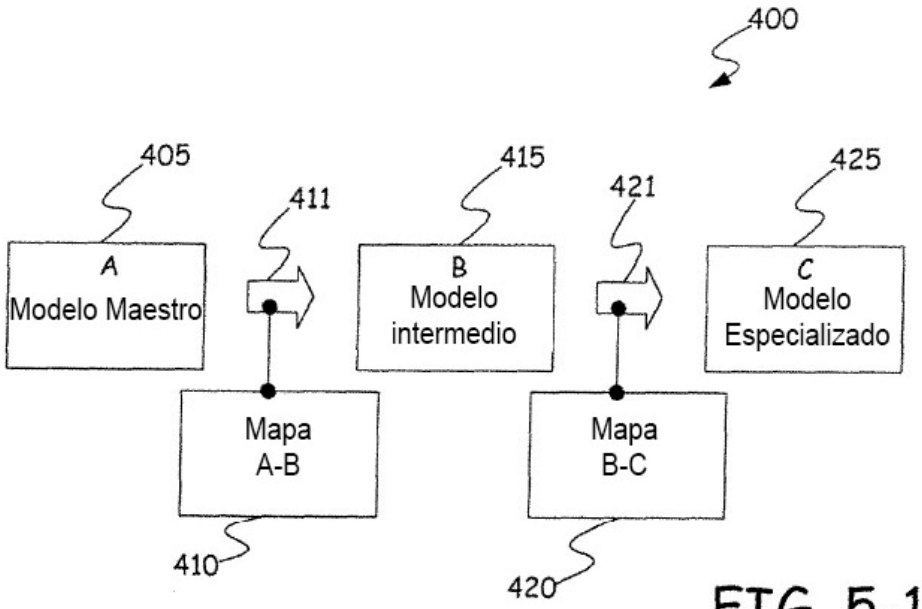
400
405
411
415
421
425
A
Modelo Maestro
B
Modelo intermedio
C
Modelo Especializado
Mapa A-B
Mapa B-C
410
420

FIG. 5-1

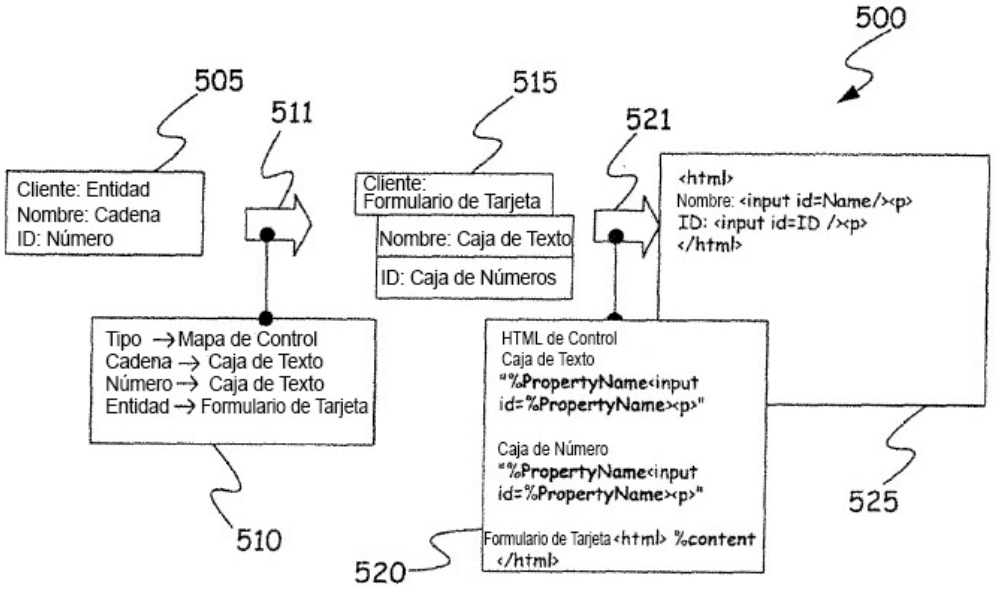
500
505
511
515
521
Cliente: Entidad
Nombre: Cadena
ID: Número
Cliente:
Formulario de Tarjeta
Nombre: Caja de Texto
ID: Caja de Números
<html>
Nombre: <input id=Name/><p>
ID: <input id=ID /><p>
</html>
Tipo → Mapa de Control
Cadena → Caja de Texto
Número → Caja de Texto
Entidad → Formulario de Tarjeta
HTML de Control
Caja de Texto
"%PropertyName<input
id=%PropertyName><p>"
Caja de Número
"%PropertyName<input
id=%PropertyName><p>"
Formulario de Tarjeta<html> %content
</html>
510
520
525

FIG. 5-3

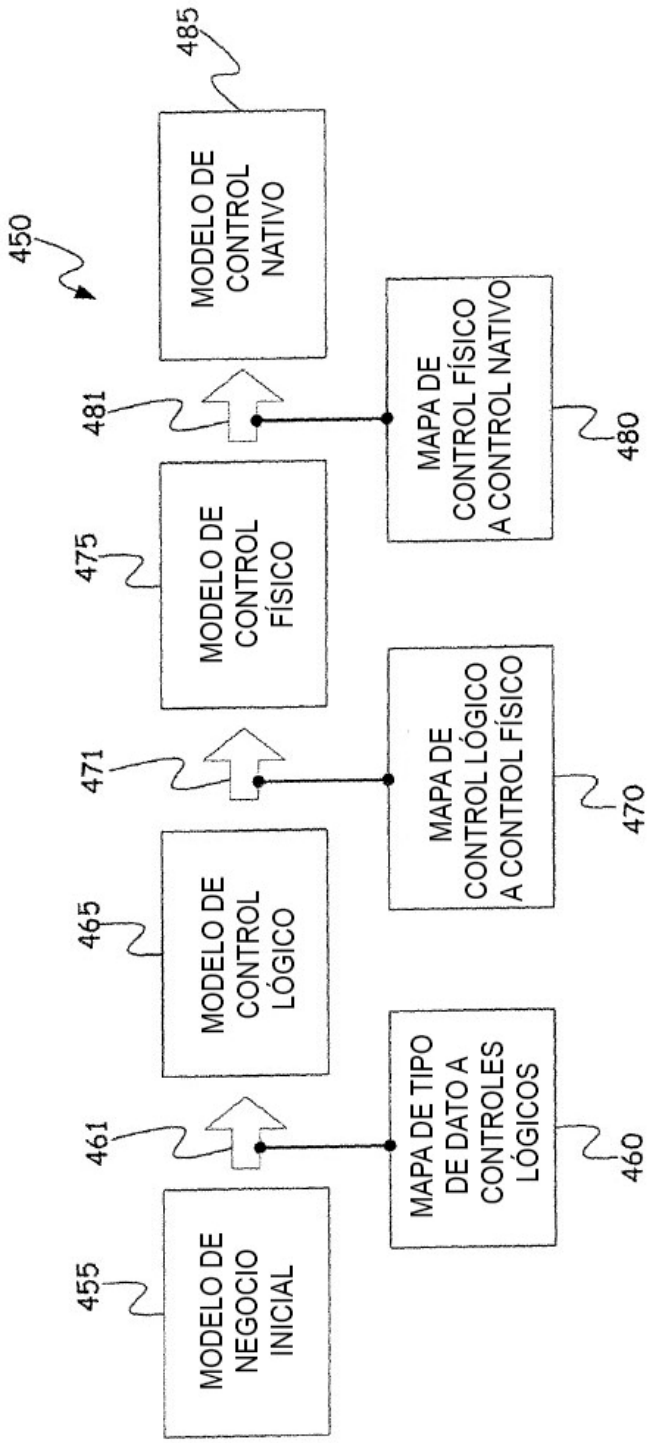
450
MODELO DE NEGOCIO INICIAL
455
461
MODELO DE CONTROL LÓGICO
465
471
MODELO DE CONTROL FÍSICO
475
481
MODELO DE CONTROL NATIVO
485
MAPA DE TIPO DE DATO A CONTROLES LÓGICOS
460
MAPA DE CONTROL LÓGICO A CONTROL FÍSICO
470
MAPA DE CONTROL FÍSICO A CONTROL NATIVO
480

FIG. 5-2

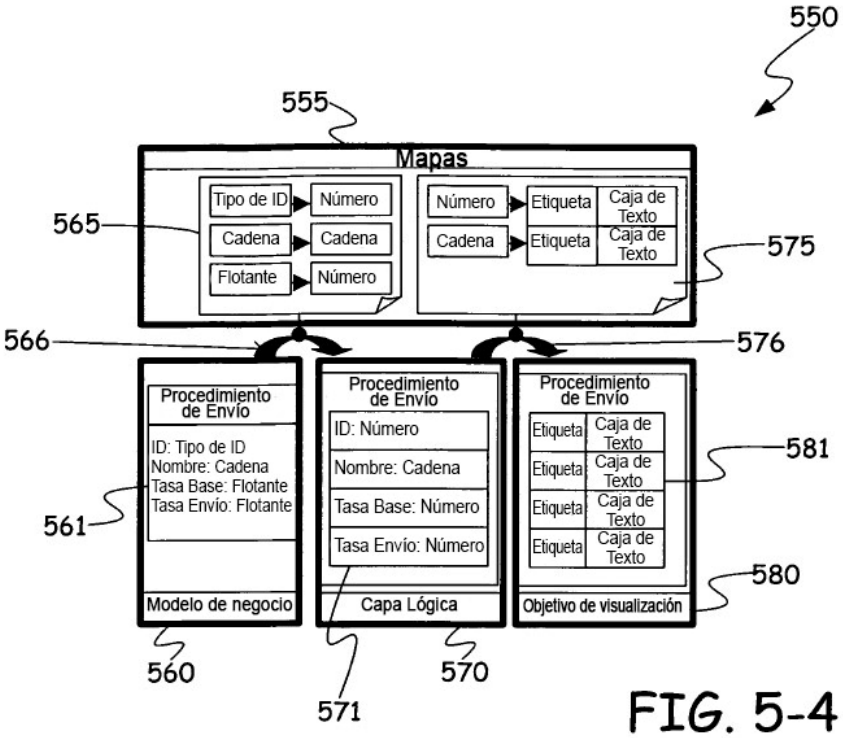
550
555
Mapas
565
Tipo de ID
Número
Cadena
Cadena
Flotante
Número
Número
Etiqueta
Caja de Texto
Cadena
Etiqueta
Caja de Texto
575
566
576
Procedimiento de Envío
ID: Tipo de ID
Nombre: Cadena
Tasa Base: Flotante
Tasa Envío: Flotante
561
Modelo de negocio
Procedimiento de Envío
ID: Número
Nombre: Cadena
Tasa Base: Número
Tasa Envío: Número
Capa Lógica
Procedimiento de Envío
Etiqueta
Caja de Texto
581
580
Objetivo de visualización
560
571
570

FIG. 5-4

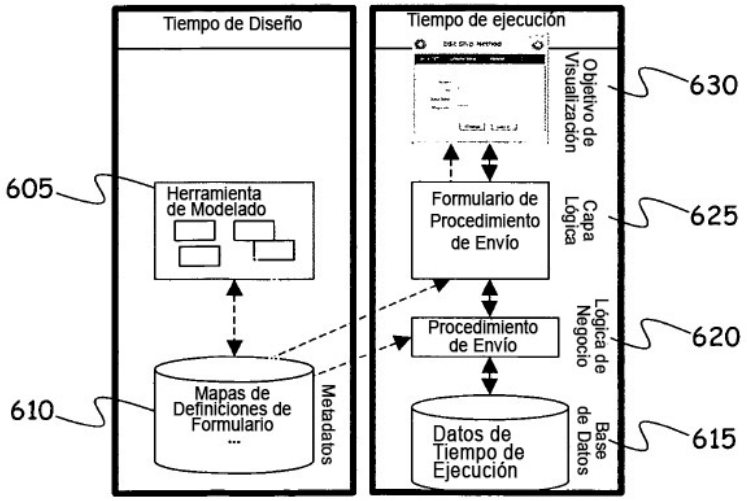
Tiempo de Diseño
Tiempo de ejecución
605
Herramienta de Modelado
610
Mapas de Definiciones de Formulario ...
Metadatos
Objetivo de Visualización
630
Formulario de Procedimiento de Envío
Capa Lógica
625
Procedimiento de Envío
Lógica de Negocio
620
Datos de Tiempo de Ejecución
Base de Datos
615

FIG. 6

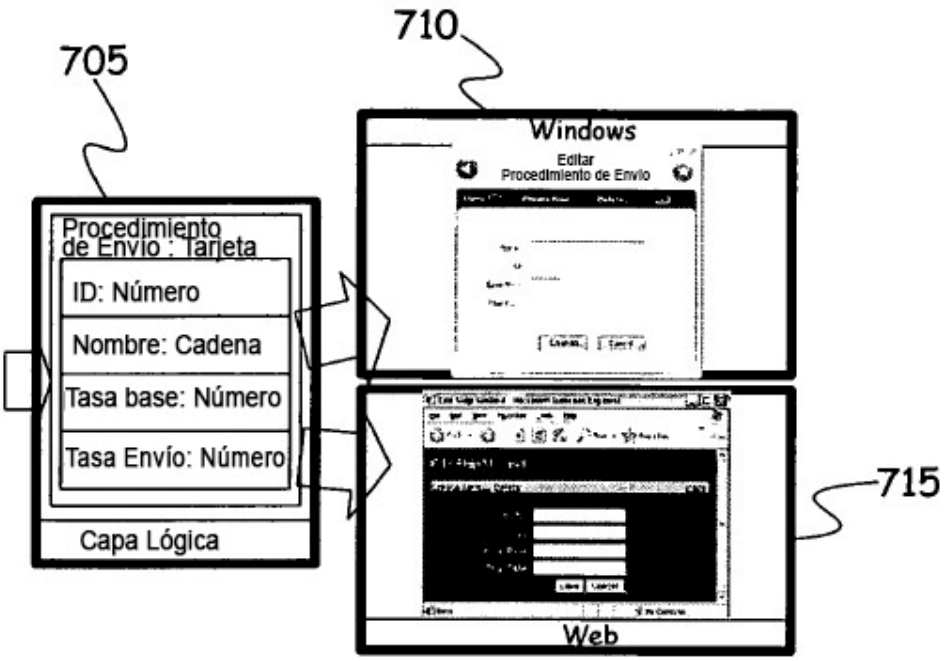
705
710
715
Procedimiento de Envío : Tarjeta
ID: Número
Nombre: Cadena
Tasa base: Número
Tasa Envío: Número
Capa Lógica
Windows
Editar Procedimiento de Envío
Web

FIG. 7

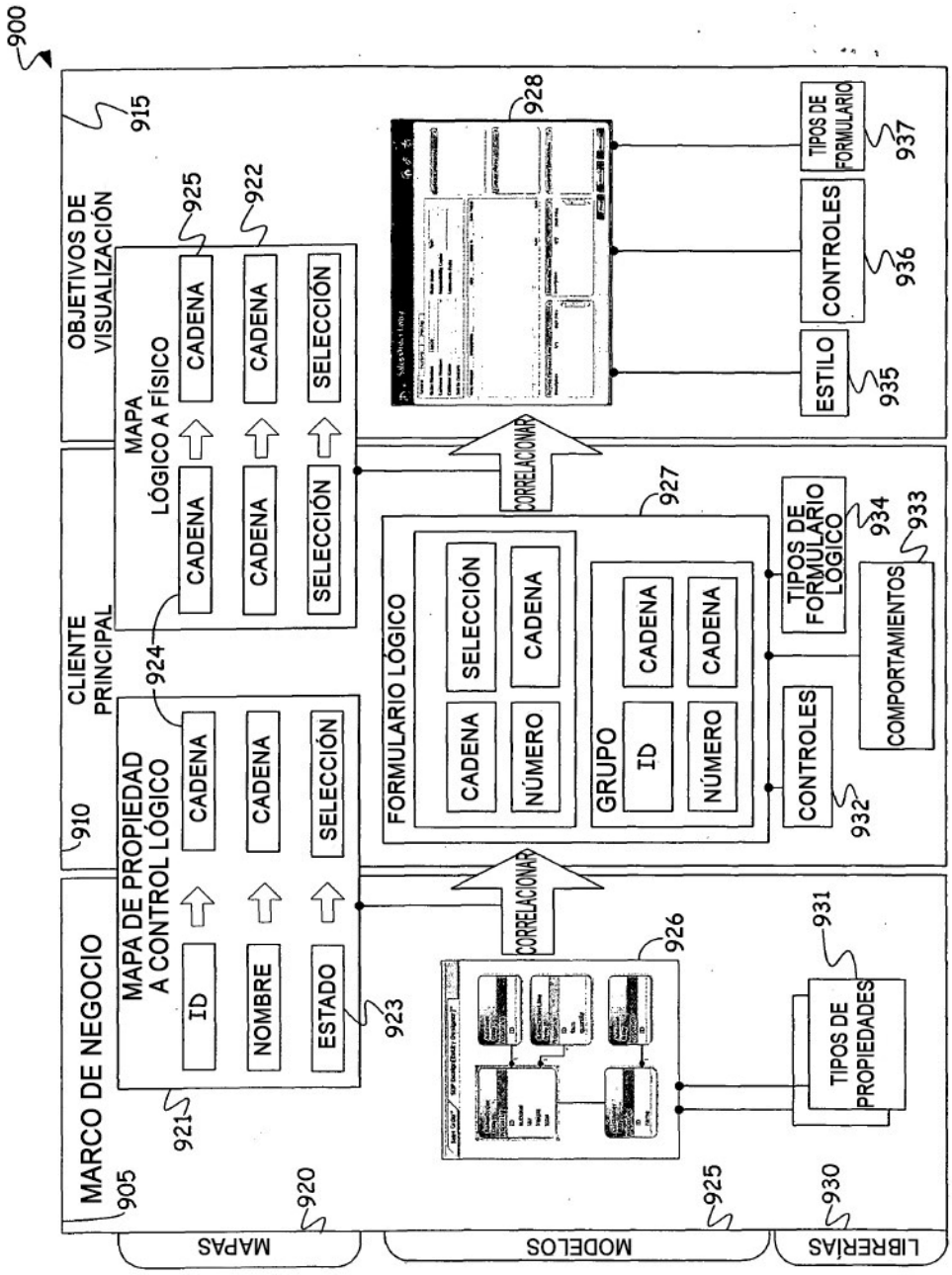
900
MARCO DE NEGOCIO
905
MAPAS
920
MODELOS
925
LIBRERÍAS
930
MAPA DE PROPIEDAD A CONTROL LÓGICO
921
ID
NOMBRE
ESTADO
923
CADENA
CADENA
SELECCIÓN
924
CLIENTE PRINCIPAL
910
MAPA LÓGICO A FÍSICO
CADENA
CADENA
SELECCIÓN
925
922
OBJETIVOS DE VISUALIZACIÓN
915
CORRELACIONAR
CORRELACIONAR
926
TIPOS DE PROPIEDADES
931
FORMULARIO LÓGICO
CADENA
SELECCIÓN
CADENA
NÚMERO
GRUPO
ID
CADENA
NÚMERO
CADENA
927
CONTROLES
932
TIPOS DE FORMULARIO LÓGICO
934
COMPORTAMIENTOS
933
928
ESTILO
935
CONTROLES
936
TIPOS DE FORMULARIO
937

FIG. 8