(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 676 739**

(51) Int. CI.:

G06F 9/445 (2008.01)
G06F 9/44 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **05.04.2005 E 05102665 (6)**

(97) Fecha y número de publicación de la concesión europea: **23.05.2018 EP 1586994**

(54) Título: **Sistema y procedimiento para enlace dinámico de controles y comandos de interfaz de usuario**

(30) Prioridad:

13.04.2004 US 822910

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
24.07.2018

(73) Titular/es:

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

(72) Inventor/es:

JENNI, DAVID J.;
COOPER, KENNETH B.;
ROEDER, LUTZ;
GUPTA, NAMITA;
BENT, SAMUEL W. y
PETERS, TED A.

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Sistema y procedimiento para enlace dinámico de controles y comandos de interfaz de usuario

Campo de la invención

La invención se refiere a un procesamiento informático y, en particular, al uso de un mecanismo de enlace de datos para enlazar comandos.

Antecedentes de la invención

El código para los programas de aplicación se separa con frecuencia en una capa para la interacción del usuario (un conjunto de "vistas" o interfaces de usuario (IU)) y una capa para implementar la lógica de aplicación interna y la gestión de datos (a menudo realizadas a través de un conjunto de "modelos"). La interfaz de usuario en general incluye elementos de menú y otros elementos de IU que invocan la funcionalidad implementada en los modelos.

Normalmente, un diseñador gráfico diseña el aspecto de la interfaz de usuario mientras que un desarrollador escribe el código que implementa la interfaz de usuario y/o el modelo subyacente. El diseño gráfico y el desarrollo de software son dos disciplinas muy diferentes y con frecuencia es difícil para los diseñadores y los desarrolladores trabajar juntos de manera productiva. Normalmente, un diseñador usa herramientas gráficas como Adobe® Photoshop® y Adobe® Illustrator® para crear una maqueta de una interfaz de usuario (IU) y a continuación un desarrollador implementa nuevamente la IU en el código. Los elementos de diseño gráfico originales no se vuelven a usar en general en la implementación final y, a veces, partes del diseño se pierden en el procedimiento debido a que el desarrollador no puede recrear fácilmente el diseño en código o no entiende completamente el diseño. Si se modifica el diseño, puede requerirse que el diseñador vuelva a dibujar la interfaz de usuario y el desarrollador tenga que reescribir partes del código para que coincidan con el diseño. En resumen, el procedimiento es tosco.

Otro problema descubierto cuando se diseña una aplicación de este tipo es cómo exponer la funcionalidad (comandos) y cómo asignar o enlazar esa funcionalidad con los elementos de la IU. Por ejemplo, una aplicación de procesamiento de texto puede exponer la funcionalidad para cortar texto seleccionado. Esta funcionalidad puede exponerse como un "comando" (por ejemplo, el documento expone una forma de invocar un comando "cortar"). Se requiere un código de programa para establecer la conexión entre lo que sucede en la IU y lo que sucede con los datos subyacentes, es decir, definir cómo se enlaza el comando al elemento de menú.

Tradicionalmente, se han empleado manejadores de eventos o encaminadores de comandos para realizar esta tarea. Los manejadores de eventos proporcionan una forma directa de enlazar elementos de código y de IU. Un elemento de IU puede exponer una instrucción de evento (por ejemplo, "hacer clic" en un elemento de menú) y el modelo puede implementar un procedimiento (por ejemplo, un manejador de eventos) que coincida con la firma de la instrucción de evento. Sin embargo, este mecanismo no es muy flexible. Los cambios en el modelo de datos activo (por ejemplo, el documento activo en el ejemplo anterior) o en el estado de comando (por ejemplo, el comando "cortar" pasa a estar inactivo o deshabilitado) normalmente requieren un código adicional escrito por un desarrollador para conectar y desconectar los manejadores de eventos y para actualizar el estado de la interfaz de usuario.

Un ejemplo de lo anterior es el documento US-B1-6 330 006 que se refiere a un procedimiento y aparato para sincronizar los objetos de interfaz de una interfaz de usuario gráfica de aplicación con los datos subyacentes. En esto, se usa una herramienta de diseño para especificar las propiedades de enlace de un objeto de interfaz. Un enlace de expresión asocia el objeto de interfaz a una pluralidad de objetos de interfaz y/o fuentes de datos subyacentes que contienen datos utilizados en la evaluación de una expresión, cuyo resultado se muestra en el objeto de interfaz. Un enlace de expresión de consulta asocia una pluralidad de objetos de interfaz y/o fuentes de datos subyacentes a una expresión evaluable, cuyo resultado se usa en una consulta para recuperar datos de las fuentes de datos. Cuando se produce un cambio en los objetos de interfaz o en los datos subyacentes se notifica a un gestor de enlaces y se procesa la aplicación de cambio para garantizar que los objetos de interfaz y/o las fuentes de datos vinculados permanezcan sincronizados.

Las aplicaciones avanzadas que requieren flexibilidad en manejar el comando normalmente asignan un identificador (ID) para cada comando asociado con un elemento de interfaz de usuario. Tales sistemas suelen tener un servicio central (a veces llamado gestor de comandos) que asigna un ID de comando a la implementación actualmente activa del comando. El gestor de comandos maneja un conjunto de destinos de comando activos (es decir, un conjunto de modelos que exponen comandos). Para ejecutar un comando, se envía un identificador desde la IU al gestor de comandos y el gestor de comandos encuentra el destino de comando que está manejando un comando con el ID recibido e invoca el comando en ese destino. Desafortunadamente, este mecanismo es bastante complejo y difícil de controlar.

Sería útil que hubiera una manera fácil y flexible para conectar elementos de IU a los comandos expuestos en el modelo de aplicación. Además, sería útil si también fuera posible facilitar el procedimiento de desarrollo de software para diseñadores y desarrolladores.

Sumario de la invención

Un motor de enlace de datos se usa para realizar el enlace de comandos. Un motor de enlace de datos enlaza los controles de interfaz de usuario tales como botones, menús, cuadros de lista, etc. a los comandos expuestos en un modelo de aplicación. En algunas realizaciones de la invención, el enlace de comandos se especifica de manera declarativa en el lenguaje de marcado. Se puede especificar una ruta de enlace de datos y una fuente de datos en el marcado que define cómo se vinculan los comandos a los elementos de IU.

Breve descripción de los dibujos

El sumario anterior, así como la siguiente descripción detallada de las realizaciones ilustrativas, se comprenderán mejor cuando se lean junto con los dibujos adjuntos. Con el fin de ilustrar la invención, en los dibujos se muestran construcciones a modo de ejemplo de la invención; sin embargo, la invención no se limita a los procedimientos e instrumentos específicos desvelados. En los dibujos:

la figura 1 es un diagrama de bloques que muestra un entorno informático a modo de ejemplo en el que pueden implementarse aspectos de la invención;
la figura 2 es un diagrama de bloques de un sistema a modo de ejemplo para el enlace de comandos de acuerdo con una realización de la invención;
la figura 3 es un diagrama de bloques más detallado de la figura 2 de acuerdo con una realización de la invención;
la figura 4 es un diagrama de bloques de un uso a modo de ejemplo del sistema de enlace de comandos de la figura 3 de acuerdo con una realización de la invención;
la figura 5 es un diagrama de flujo de un procedimiento a modo de ejemplo del enlace de comandos de acuerdo con una realización de la invención; y
la figura 6 es un diagrama de bloques de objetos de acuerdo con un aspecto de la invención.

Descripción detallada de las realizaciones ilustrativas

Visión de conjunto

Supongamos que una aplicación de procesamiento de texto maneja un número de objetos de documento que exponen la funcionalidad para el corte de texto seleccionado. La aplicación puede proporcionar un elemento de menú con la etiqueta “Cortar” en el menú “Editar”. La aplicación necesita definir cómo el comando “Cortar” se enlaza al elemento de menú y al documento activo. Los procedimientos tradicionales de enlace de comandos son complejos y se suman a las dificultades inherentes a la interacción diseñador/desarrollador. De acuerdo con algunas realizaciones de la invención, el enlace de comandos se realiza asociando de manera declarativa las rutas de comandos de enlace de datos a elementos o componentes de una interfaz de usuario, enviando notificaciones de cambio cuando una propiedad de comando cambia y actualizando automáticamente el destino mediante un mecanismo de enlace de objetos, disminuyendo de este modo el nivel de complejidad y la necesidad de experiencia técnica en enlazar y encaminar comandos.

Entorno informático a modo de ejemplo

La figura 1 y la siguiente exposición pretenden proporcionar una breve descripción general de un entorno informático adecuado en el que puede implementarse la invención. Debería entenderse, sin embargo, que se contemplan dispositivos de mano, portátiles y dispositivos informáticos de todo tipo para su uso en relación con la presente invención. Aunque a continuación se describe un ordenador de fin general, este no es más que un ejemplo, y la presente invención requiere solo un cliente ligero que tenga interoperabilidad e interacción con el servidor de red. Por lo tanto, la presente invención puede implementarse en un entorno de servicios alojados en red en el que están implicados muy pocos o mínimos recursos de cliente, por ejemplo, un entorno de red en el que el dispositivo cliente sirve simplemente como un navegador o una interfaz para la red informática mundial.

Aunque no se requiere, la invención puede implementarse a través de una interfaz de programación de aplicaciones (API), para su uso por un desarrollador, y/o incluido en el software de navegación de red que se describirá en el contexto general de las instrucciones ejecutables por ordenador, tales como módulos de programa, que se ejecutan en uno o más ordenadores, tales como estaciones de trabajo de cliente, servidores u otros dispositivos. En general, los módulos de programa incluyen rutinas, programas, objetos, componentes, estructuras de datos y similares que realizan tareas específicas o implementan tipos de datos abstractos específicos. Normalmente, la funcionalidad de los módulos de programa puede combinarse o distribuirse como se desee en diversas realizaciones. Además, los expertos en la materia apreciarán que la invención puede practicarse con otras configuraciones de sistemas informáticos. Otros sistemas informáticos, entornos y/o configuraciones bien conocidos que pueden ser adecuados para su uso con la invención incluyen, pero no se limitan a, ordenadores personales (PC), cajeros automáticos, ordenadores de servidor, dispositivos de mano o portátiles, sistemas de procesadores múltiples, sistemas basados en microprocesador, electrónica de consumo programable, PC en red, minicomputadoras, ordenadores centrales y similares. La invención también puede ponerse en práctica en entornos informáticos distribuidos en los que las tareas se realizan mediante dispositivos de procesamiento remoto que están enlazados a través de una red de comunicaciones u otro medio de transmisión de datos. En un entorno informático distribuido, los módulos de

programa pueden localizarse tanto en medios de almacenamiento informáticos locales como remotos, incluidos los dispositivos de almacenamiento de memoria.

Por lo tanto, la figura 1 ilustra un ejemplo de un entorno 100 de sistema informático adecuado en el que puede implementarse la invención, aunque como se ha aclarado anteriormente, el entorno 100 de sistema informático es solo un ejemplo de un entorno informático adecuado y no pretende sugerir ninguna limitación en cuanto al ámbito de uso o la funcionalidad de la invención. Tampoco debería interpretarse que el entorno 100 informático tiene alguna dependencia o requisito relacionado con uno cualquiera o una combinación de componentes ilustrados en el entorno 100 de operación a modo de ejemplo.

Haciendo referencia a la figura 1, un sistema a modo de ejemplo para implementar la invención incluye un dispositivo informático de fin general en la forma de un ordenador 110. Los componentes del ordenador 110 pueden incluir, pero no están limitados a, una unidad 120 de procesamiento, una memoria 130 de sistema y un bus 121 de sistema que acopla diversos componentes del sistema incluyendo la memoria de sistema a la unidad 120 de procesamiento. El bus 121 de sistema puede ser cualquiera de diversos tipos de estructuras de bus incluyendo un bus de memoria o un controlador de memoria, un bus periférico y un bus local que usa cualquiera de una variedad de arquitecturas de bus. A modo de ejemplo, y no de limitación, tales arquitecturas incluyen el bus de arquitectura estándar industrial (ISA), el bus de arquitectura de micro canal (MCA), el bus ISA mejorado (EISA), el bus local de asociación para estándares electrónicos y de video (VESA) y el bus de interconexión de componentes periféricos (PCI) (también conocido como bus Mezzanine).

El ordenador 110 incluye normalmente una variedad de medios legibles por ordenador. Los medios legibles por ordenador pueden ser cualquier medio disponible al que pueda accederse mediante el ordenador 110 e incluye tanto medios volátiles como no volátiles, medios removibles como no removibles. A modo de ejemplo, y no de limitación, los medios legibles por ordenador pueden comprender medios de almacenamiento informático y medios de comunicación. Los medios de almacenamiento informático incluyen medios volátiles y no volátiles, removibles y no removibles implementados en cualquier procedimiento o tecnología para el almacenamiento de información tal como instrucciones legibles por ordenador, estructuras de datos, módulos de programa u otros datos. Los medios de almacenamiento informático incluyen, pero no están limitados a memoria RAM, ROM, EEPROM, flash u otra tecnología de memoria, CDROM, discos versátiles digitales (DVD) u otro tipo de almacenamiento en disco óptico, casetes magnéticos, cinta magnética, almacenamiento en disco magnético u otros dispositivos magnéticos de almacenamiento, o cualquier otro medio que pueda usarse para almacenar la información deseada y a la que se pueda acceder por el ordenador 110. Los medios de comunicación incorporan en general 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 entrega 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 codifican información en la señal. A modo de ejemplo, y no de limitación, los medios de comunicación incluyen medios cableados tales como una red cableada o una conexión de cableado directo, y medios inalámbricos tales como medios acústicos, de RF, infrarrojos y otros medios inalámbricos. Las combinaciones de cualquiera de los anteriores también deben incluirse dentro del ámbito de los medios legibles por ordenador.

La memoria 130 de sistema incluye medios de almacenamiento informático en la forma de memoria volátil y/o no volátil tal como una memoria 131 de solo lectura (ROM) y una memoria 132 de acceso aleatorio (RAM). Un sistema 133 de entrada/salida básico (BIOS), que contiene las rutinas básicas que ayudan a transferir información entre elementos dentro del ordenador 110, tal como durante la puesta en marcha, normalmente se almacenan en la ROM 131. La RAM 132 contiene normalmente datos y/o módulos de programas que pueden accederse inmediatamente y/o están operándose actualmente por la unidad 120 de procesamiento. A modo de ejemplo, y no de limitación, la figura 1a ilustra el sistema 134 operativo, los programas 135 de aplicación, otros módulos 136 de programa y los datos 137 de programa.

El ordenador 110 también puede incluir otros medios removibles/no removibles, volátiles/no volátiles de almacenamiento informático. A modo de ejemplo solamente, la figura 1a ilustra una unidad 141 de disco duro que lee o escribe en medios magnéticos no volátiles no removibles, una unidad 151 de disco magnético que lee o escribe en un disco 152 magnético no volátil removible, y una unidad 155 de disco óptico que lee o escribe en un disco 156 óptico no volátil removible, tal como un CDROM u otro medio óptico. Otros medios de almacenamiento informático volátiles/no volátiles removibles/no removibles que pueden usarse en el entorno operativo a modo de ejemplo incluyen, pero no se limitan a, casetes de cinta magnética, tarjetas de memoria flash, discos versátiles digitales, cintas de video digital, RAM de estado sólido, ROM de estado sólido, y similares. La unidad 141 de disco duro se conecta normalmente al bus 121 de sistema a través de una interfaz de memoria no removible tal como la interfaz 140, y la unidad 151 de disco magnético y la unidad 155 de disco óptico se conectan normalmente al bus 121 de sistema mediante una interfaz de memoria removible, tal como la interfaz 150.

Las unidades y sus medios de almacenamiento informático asociados tratados anteriormente e ilustrados en la figura 1 proporcionan un almacenamiento de instrucciones legibles por ordenador, estructuras de datos, módulos de programa y otros datos al ordenador 110. En la figura 1, por ejemplo, la unidad 141 de disco duro se ilustra como sistema 144 operativo de almacenamiento, programas 145 de aplicación, otros módulos 146 de programa y datos

147 de programa. Obsérvese que estos componentes pueden, o ser los mismos o diferentes del sistema 134 operativo, los programas 135 de aplicación, los otros módulos 136 de programa, y los datos 137 de programa. El sistema 144 operativo, los programas 145 de aplicación, los otros 146 módulos de programa y los datos 147 de programa reciben diferentes números en este caso para ilustrar que, como mínimo, son copias diferentes. Un usuario puede introducir comandos e información en el ordenador 110 a través de dispositivos de entrada tales como un teclado 162 y un dispositivo 161 señalador, comúnmente referido como un ratón, una bola de seguimiento o un teclado táctil. Otros dispositivos de entrada (no mostrados) pueden incluir un micrófono, una palanca de mando, una almohadilla para juegos, una antena parabólica, un escáner o similares. Estos y otros dispositivos de entrada a menudo están conectados a la unidad 120 de procesamiento a través de una interfaz 160 de entrada de usuario que está acoplada al bus 121 de sistema, pero puede conectarse mediante otras estructuras de interfaz y bus, tales como un puerto paralelo, un puerto de juegos o bus serie universal (USB).

Un monitor 191 u otro tipo de dispositivo de visualización también está conectado al bus 121 de sistema a través de una interfaz, tal como una interfaz 190 de vídeo. Una interfaz 182 gráfica, tal como el puente norte, también puede estar conectada al bus 121 de sistema. El puente norte es un conjunto de chips que se comunica con la CPU, o la unidad 120 de procesamiento de host, y asume la responsabilidad de las comunicaciones del puerto de gráficos acelerados (AGP). Una o más unidades 184 de procesamiento de gráficos (GPU) pueden comunicarse con la interfaz 182 gráfica. A este respecto, las GPU 184 incluyen en general un almacenamiento de memoria en chip, tal como el almacenamiento de registros y las GPU 184 se comunican con una memoria 186 de video. Las GPU 184, sin embargo, no son más que un ejemplo de un coprocesador y por lo tanto, pueden incluirse una variedad de dispositivos de coprocesamiento en el ordenador 110. 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, que a su vez puede comunicarse con la memoria 186 de video. Además del monitor 191, los ordenadores también pueden incluir otros dispositivos de salida periféricos tales como unos altavoces 197 y una impresora 196, que pueden estar conectados a través de una interfaz 195 periférica de salida.

El ordenador 110 puede operar en un entorno de red usando conexiones lógicas a uno o más ordenadores remotos, tales como un ordenador 180 remoto. El ordenador 180 remoto puede ser un ordenador personal, un servidor, un encaminador, un PC de red, un dispositivo del mismo nivel u otro nodo de red común, y normalmente incluye muchos o todos los elementos descritos anteriormente con respecto al ordenador 110, aunque en la figura 1a se ha ilustrado solamente un dispositivo 181 de almacenamiento de memoria. Las conexiones lógicas representadas en la figura 1a incluyen una red 171 de área local (LAN) y una red 173 de área extensa (WAN), pero también puede incluir otras redes. Dichos entornos de red son comunes en oficinas, redes informáticas de toda la empresa, intranets e Internet.

Cuando se usa en un entorno de red LAN, el ordenador 110 está conectado a la LAN 171 a través de una interfaz de red o adaptador 170. Cuando se usa en un entorno de red WAN, el ordenador 110 incluye normalmente un módem 172 u otros medios para establecer 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 de red, los módulos de programa representados con respecto al ordenador 110, o partes de los mismos, pueden almacenarse en el dispositivo de almacenamiento de memoria remota. A modo de ejemplo, y no de limitación, la figura 1a ilustra unos programas 185 de aplicación remota que residen en el dispositivo 181 de memoria. Se apreciará que las conexiones de red mostradas son a modo de ejemplo y pueden usarse otros medios para establecer un enlace de comunicaciones entre los ordenadores.

Un experto en la materia puede apreciar que un ordenador 110 u otro dispositivo cliente puede desplegarse como parte de una red informática. A este respecto, la presente invención se refiere a cualquier sistema informático que tenga cualquier cantidad de unidades de memoria o almacenamiento, y a cualquier cantidad de aplicaciones y procedimientos que se produzcan a través de cualquier cantidad de unidades o volúmenes de almacenamiento. La presente invención puede aplicarse a un entorno con ordenadores servidores y ordenadores cliente desplegados en un entorno de red, teniendo un almacenamiento remoto o local. La presente invención también puede aplicarse a un dispositivo informático independiente, que tenga capacidades de funcionalidad de lenguaje de programación, interpretación y ejecución.

Sistema y procedimiento para usar un mecanismo de enlace de datos para enlazar comandos

La figura 2 ilustra un sistema a modo de ejemplo para usar un mecanismo de enlace de datos para realizar un enlace de comandos de acuerdo con algunas realizaciones de la invención. Un sistema de este tipo puede residir en su totalidad o en parte en uno o más ordenadores, tal como el ordenador 202 a modo de ejemplo de la figura 2. El ordenador 202 puede comprender un ordenador tal como el ordenador 110 descrito con respecto a la figura 1. Un sistema para usar un motor de enlace de datos para encaminar comandos puede comprender uno o más de los siguientes elementos: un componente 208 de enlace de datos, una fuente 206 y un destino 203.

En algunas realizaciones de la invención, el componente 208 de enlace de datos es un motor 208 de enlace de datos, que permite el enlace dinámico de un objeto 212a de comando de un objeto de fuente, (por ejemplo, los objetos 210a, 210b, etc. de fuente a modo de ejemplo) a un objeto de destino, (por ejemplo, los objetos 204a, 204b, etc. destino a modo de ejemplo). El motor de enlace de datos puede escuchar las notificaciones de cambio de

propiedad en los objetos, de tal manera que un cambio en una propiedad de comando de objeto de fuente se refleje automáticamente en la propiedad de objeto de destino asociada. El motor de enlace de datos puede escuchar las notificaciones de cambio de propiedad en los objetos, de tal manera que un cambio en una propiedad no de comando de objeto de fuente se refleje automáticamente en la propiedad de objeto de destino asociada y viceversa. Un objeto de destino puede estar asociado a una fuente de datos, que identifica la fuente a la que está vinculado el objeto de destino. El motor de enlace de datos puede soportar la evaluación de las rutas de propiedad para permitir el enlace de partes específicas del destino a partes específicas de la fuente. En algunas realizaciones de la invención, enlazar las propiedades de objeto de destino a las propiedades de comando de objeto de fuente puede hacerse de manera declarativa en un lenguaje de marcado tal como HTML (lenguaje de marcado de hipertexto), XML (lenguaje de marcado extensible), XAML (lenguaje de marcado de aplicación extensible) u otro lenguaje de marcado adecuado. El motor de enlace de datos puede buscar la propiedad de comando de objeto de fuente en la fuente de datos del objeto de destino y realizar la actualización adecuada.

En algunas realizaciones de la invención, se genera una gráfica de objetos orientados a objetos donde uno, algunos, o todos los objetos apuntan a otros objetos, formando una gráfica donde cada flecha que apunta desde un objeto a otro en la gráfica representa una propiedad. Una gráfica a modo de ejemplo de objetos orientados a objetos se ilustra en la figura 6. En la figura 6, el objeto 602 representa un objeto de gestor de documento, los objetos 604 y 606 representan objetos de documento y el objeto 608 representa un comando de “corte”. En la figura 6, el documento activo es el objeto de documento 606 (indicado por la línea sólida 603 que representa la propiedad ActiveDocument). Se entenderá que la invención tal como se contempla no está limitada a tales objetos y comandos. Cualquier objeto y comando adecuado pueden actuar en consecuencia. De hecho, la invención tal como se contempla no se limita a objetos en un entorno de lenguaje de programación orientado a objetos, sino que puede aplicarse igualmente a cualquier fuente de datos y jerarquía de propiedades. En algunas realizaciones de la invención, el motor 208 de enlace de datos permite la especificación de una fuente de datos (objeto 602) y una ruta de propiedad tal como “ActiveDocument.CutCommand” que representa la ruta del objeto 602 al objeto 606 al objeto 608. El motor 208 de enlace de datos puede consultar en la gráfica de objetos qué representan los objetos “activos” dentro de un programa en ejecución, para determinar dinámicamente qué objeto está representado por la ruta “ActiveDocument.CutCommand” (en este caso, el objeto 608).

Haciendo referencia de nuevo a la figura 2, la fuente 206 puede incluir uno o más objetos fuente como los representados por los objetos 210a, 210b, etc. de fuente. Los objetos 210a, 210b, etc. de fuente pueden estar asociados con uno o más objetos de comando de fuente, representados en la figura 2 por el objeto 212a, etc. de comando de fuente. En algunas realizaciones de la invención, la fuente 206 puede representar un modelo. Un modelo en algunas realizaciones de la invención es la lógica de aplicación subyacente que representa una recopilación del estado subyacente. Por ejemplo, considérese una aplicación que permite a un usuario explorar el sistema de archivos. El modelo para la aplicación en este caso puede ser un sistema de archivos: el conjunto de carpetas y archivos dentro de las carpetas de un directorio seleccionado. Un número de vistas pueden estar vinculadas a un modelo. Las vistas vinculadas al modelo pueden depender del modelo. En algunas realizaciones de la invención, sin embargo, el modelo no depende de la vista o las vistas. Un modelo puede enviar una notificación de cambio si una propiedad en uno de sus objetos cambia o si se produce un cambio de estado. Por ejemplo, si se agrega un archivo nuevo a una carpeta, puede enviarse una notificación de cambio.

Los objetos 210a, 210b, etc. de fuente pueden estar asociados con uno o más objetos de comando como se representa por los objetos 212a, etc. de comando de fuente. En algunas realizaciones de la invención, un objeto de comando es un objeto que está asociado con un procedimiento de ejecución, (es decir, un objeto de comando es un objeto ejecutable) y tiene estado. El estado asociado con el objeto de comando en algunas realizaciones de la invención es un valor booleano que representa si el comando puede o no ejecutarse, es decir, si el comando está o no activo (habilitado) o inactivo (no habilitado). Los ejemplos de comandos incluyen pero no se limitan a “abrir un documento”, “cortar el texto seleccionado”, y así sucesivamente.

El destino 203 puede incluir uno o más objetos de destino, como se representa en la figura 2 por los objetos 204a, 204b, etc. de destino. En algunas realizaciones de la invención, el destino puede ser una vista o una interfaz de usuario. Una o más vistas pueden estar vinculadas a un modelo y mostrar el estado del modelo subyacente. En algunas realizaciones de la invención, una vista o una interfaz de usuario se define en un lenguaje de marcado tal como HTML (lenguaje de marcado de hipertexto), XML (lenguaje de marcado extensible), XAML (lenguaje de marcado de aplicación extensible) u otro lenguaje de marcado adecuado, en el que el aspecto de la interfaz de usuario está definido y los elementos o componentes de la interfaz de usuario están definidos. Un objeto de destino puede ser un elemento o control de interfaz de usuario tal como, pero no limitado a, un elemento de menú, botón o cuadro de lista. En el ejemplo de sistema de archivos descrito anteriormente, una interfaz de usuario a modo de ejemplo puede mostrar la lista de archivos actuales en las carpetas del directorio seleccionado.

Para enlazar la interfaz de usuario al modelo subyacente, en algunas realizaciones de la invención, en lugar de definir explícitamente el enlace usando un manejador de eventos o indirectamente asociando el componente de interfaz de usuario al modelo subyacente a través de la asignación de un ID e invocar un gestor de comando, el objeto que representa el componente de interfaz de usuario se vincula al objeto de modelo subyacente especificando un objeto de fuente de datos y una ruta de consulta, como se ha descrito anteriormente con respecto a la figura 6. Si alguna parte de la consulta cambia, el objeto envía una notificación de cambio y el motor de enlace

de datos detecta la notificación de cambio y actualiza el objeto(s) correspondiente. Se apreciará que el sujeto que se está consultando son objetos activos dentro de un programa en ejecución.

La figura 3 es un diagrama de bloques más detallado del sistema de la figura 2. En la figura 3, el motor 320 de enlace de datos puede enlazar una propiedad 322 de comando con un objeto de fuente (parte de un modelo 324) a una propiedad en un objeto de destino (por ejemplo, un componente 330 de IU). El motor 320 de enlace de datos puede escuchar las notificaciones de cambio de propiedad (notificación 326 para la propiedad 322 de modelo, notificación 328 para la propiedad 334 de IU) y sincronizar automáticamente la fuente 324 y el destino 330. Un destino 330 (por ejemplo, un componente de IU) puede estar asociado con una fuente 332 de datos que actúa como la fuente a la que está vinculado el destino 330.

En algunas realizaciones de la invención, la propiedad de IU para enlazar 334 es una propiedad receptora o destino para un objeto de comando y la propiedad de modelo para enlazar 322 es una propiedad de comando en el modelo. Una propiedad de destino a modo de ejemplo no limitativa puede ser, por ejemplo, “ClickCommand”. En algunas realizaciones de la invención, el enlace de comandos se logra asignando el modelo 324 como una fuente de datos y datos que enlazan la propiedad en el componente 334 de IU con la propiedad de comando en el modelo 322. El enlace puede lograrse de manera declarativa en algunas realizaciones, sin requerir ningún código de programa. Una instrucción declarativa a modo de ejemplo puede ser:

<MenuItem ClickCommand=“*Bind(DataSource=model,
Path=DocumentManager. ActiveDocument.CutCommand)”/>

Esta instrucción significa que la fuente de datos asociada con el elemento de menú “ClickCommand” es el objeto “modelo” y el objeto a asociarse con ClickCommand es la propiedad CutCommand en el documento activo. La IU puede crearse dentro de una herramienta de diseño de IU en el marcado sin escribir código, o fuera de una herramienta de diseño en el marcado. Como alternativa, el mismo resultado puede lograrse escribiendo el código del programa.

En algunas realizaciones de la invención, el objeto de comando puede estar asociado con el estado adicional expuesto como propiedades. Los ejemplos incluyen, pero no se limitan a IsEnabled, nombre textual a representar para el usuario, enlace clave o icono. El estado puede, o proporcionarse explícitamente por el desarrollador o calcularse u obtenerse a partir de otras propiedades de la aplicación. En algunas realizaciones de la invención, el objeto de destino sabe cómo manejar estas propiedades (por ejemplo, un elemento de menú establece su estado habilitado a la propiedad IsEnabled del objeto de comando y actualiza su aspecto visual, un elemento de menú actualiza el texto representado para el usuario a la propiedad Text en el objeto de comando). El objeto de comando proporciona notificaciones de cambio para esas propiedades. El objeto de destino escucha para cambiar las notificaciones y actualiza sus propiedades y su aspecto en el caso de una notificación de cambio. No todas las propiedades en el objeto de comando deben conocerse por el objeto de destino.

En alguna realización de la invención el objeto de destino no maneja implícitamente algunas propiedades de estado. En este caso, se usa el enlace de datos para enlazar una propiedad conocida en el objeto de comando a una propiedad conocida en el objeto de destino:

<MenuItem
Text=“*Bind (Path=DocumentManager. ActiveDocument. CutCommand. Text” />.

En alguna realización de la invención, el objeto de comando no tiene estado y no proporciona propiedades en absoluto. Las propiedades de estado aún pueden proporcionarse por el modelo y vincularse a los elementos de destino mediante el enlace de datos.

En alguna realización de la invención, los comandos sin estado pueden realizarse proporcionando un procedimiento en el modelo y enlazándose a ese procedimiento (en lugar de a un objeto de comando).

<MenuItem Click = “* Bind (Path = ActiveDocument. Copy)”
Enabled = “* Bind (Path = ActiveDocument.CanCopy”) />

En este caso no se necesita ningún objeto de comando. El modelo proporciona el estado CanCopy y el procedimiento Copy () se vincula usando el enlace de datos.

La figura 4 ilustra un escenario típico en el que puede emplearse un sistema para el enlace de comandos de acuerdo con algunas realizaciones de la invención. Considérese una aplicación que maneja múltiples documentos 408, 410, etc., uno de los cuales está activo (es decir, el documento 410). Un gestor 404 de documentos (parte del modelo) puede exponer una propiedad para el documento activo. Los objetos 408, 410, etc. de documento, (parte del modelo) pueden exponer comandos de edición como propiedades, (por ejemplo, el objeto 412 de comando) que pueden seleccionarse. Puede lograrse un enlace declarativo en algunas realizaciones usando una ruta de propiedad primero direccionando el documento 410 activo y, a continuación seleccionando un comando de edición en el documento activo a través de un botón en una interfaz 402 de usuario. Unas instrucciones de enlace declarativas a modo de ejemplo pueden ser:

<MenuItem ClickCommand =
"* Bind (Path = DocumentManager.ActiveDocument. CutCommand)" />

o:

<MenuItem ClickCommand =
"* Bind (Path = DocumentManager.ActiveDocument. CopyCommand)" />

o:

<MenuItem ClickCommand =
"* Bind (Path = DocumentManager.ActiveDocument.PasteCommand)" />

En algunas realizaciones, el gestor 404 de documentos puede proporcionar una notificación de cambio para cualquier cambio en la propiedad del documento 410 activo (por ejemplo, el usuario cambia el documento activo). La notificación de cambio se detecta por el motor 406 de enlace de datos y el comando se vuelve a vincular al documento activo. Se observará que en algunas realizaciones de la invención, el resultado anterior se realiza de manera declarativa sin la creación de un código de programa. No se requiere ningún servicio para la gestión de comandos.

En algunas realizaciones de la invención, pueden vincularse unas propiedades arbitrarias que no son propiedades de comando a unos elementos de menú. Por ejemplo, una propiedad booleana en un modelo puede estar vinculada a un elemento de menú de casilla de verificación. En algunas realizaciones de la invención, un componente de IU de casilla de verificación puede exponer una propiedad tal como "IsChecked" que puede usarse como propiedad de destino. Del mismo modo, las listas de objetos pueden vincularse a un elemento de menú de cuadro de lista especial para abordar un escenario tal como, por ejemplo, la lista de archivos "usados más recientemente".

La figura 5 ilustra un diagrama de flujo a modo de ejemplo para un procedimiento de especificar y ejecutar enlaces de comando de acuerdo con una realización de la invención. Una o más etapas del procedimiento pueden ser opcionales. En la figura 5, en la etapa 502 se recibe una definición de fuente. Esto puede implicar la codificación y/o la creación de instancias de un modelo o aplicación. Por ejemplo, un desarrollador puede implementar el modelo, exponiendo una o más propiedades a las que se vinculará. En la etapa 504, se recibe una definición de destino. Esto puede implicar la especificación en el código o el marcado de un destino, tal como una interfaz de usuario y/o la creación de instancias de la interfaz de usuario. Por ejemplo, un diseñador puede crear una interfaz de usuario en un lenguaje de marcado o un desarrollador puede crear una interfaz de usuario en código. En la etapa 506, puede recibirse una instrucción de enlace de comandos. Esto puede implicar la codificación de las conexiones de datos entre el destino y la fuente (por ejemplo, por un desarrollador) o la especificación de unas instrucciones de enlace de comandos en el marcado (por ejemplo, por un diseñador) como se ha descrito anteriormente. En algunas realizaciones de la invención, la especificación de la instrucción de enlace de comandos puede incorporarse en la definición del destino (paso 504).

En la etapa 508, la aplicación que comprende el modelo y la interfaz de usuario puede generarse y/o instanciarse. Un motor de enlace puede escuchar en las rutas del modelo y evaluar la instrucción de enlace de comandos. Si la instrucción de enlace de comando se evalúa con éxito, el destino y el modelo pueden sincronizarse como se ha descrito anteriormente.
En algunas realizaciones de la invención, puede proporcionarse una o más instrucciones de enlace de comandos. El enlace de comandos puede lograrse asignando de manera declarativa una o más rutas de comandos de enlace de datos y fuentes de datos a una fuente como se ha descrito anteriormente. En algunas realizaciones de la invención, la ruta de comando y la fuente de datos se definen de manera declarativa en un lenguaje de marcado tal como HTML, XML, XAML u otro lenguaje de marcado adecuado. Las rutas del modelo pueden monitorizarse continuamente para las notificaciones de cambio. Puede recibirse una notificación de cambio desde un objeto de comando que indica que ha cambiado una propiedad de comando. En algunas realizaciones de la invención, la notificación de cambio se envía por el objeto de comando y se detecta por un motor de enlace de datos. Si se recibe una notificación de cambio, la instrucción de enlace de comandos puede volver a evaluarse y sincronizarse la fuente y el destino como se ha descrito anteriormente. En algunas realizaciones, el motor de enlace de datos consulta en la gráfica de objetos para encontrar el objeto indicado por la fuente de datos y la ruta de datos y el motor de enlace de datos actualiza el destino automáticamente mediante un mecanismo de enlace de objetos genérico.

Las diversas técnicas descritas en el presente documento pueden implementarse junto con hardware o software o, en su caso, con una combinación de ambos. Por lo tanto, los procedimientos y aparatos de la presente invención, o ciertos aspectos o partes de los mismos, pueden adoptar la forma de código de programa (es decir, instrucciones) incorporado en medios tangibles, tales como disquetes, CD-ROM, discos duros o cualquier otro medio de almacenamiento legible por máquina, en el que, cuando el código de programa se carga y se ejecuta en una máquina, tal como un ordenador, la máquina se convierte en un aparato para poner en práctica la invención. En el caso de la ejecución de código de programa en ordenadores programables, el dispositivo informático incluirá en general un procesador, un medio de almacenamiento legible por el procesador (que incluye una memoria volátil y no volátil y/o elementos de almacenamiento), al menos un dispositivo de entrada, y en al menos un dispositivo de

salida. Uno o más programas que pueden usar la creación y/o la implementación de aspectos de modelos de programación específicos del dominio de la presente invención, por ejemplo, a través del uso de una API de procesamiento de datos o similares, se implementan preferentemente en un procedimiento de alto nivel o lenguaje de programación orientado a objetos para comunicarse con un sistema informático. Sin embargo, el programa(s) puede implementarse en ensamblador o lenguaje máquina, si se desea. En cualquier caso, el lenguaje puede ser un lenguaje compilado o interpretado, y combinado con implementaciones de hardware.

Aunque la presente invención se ha descrito junto con las realizaciones preferidas de las diversas figuras, ha de entenderse que pueden usarse otras realizaciones similares o pueden hacerse modificaciones y adiciones a las realizaciones descritas para realizar la misma función de la presente invención sin desviarse de la misma. Por lo tanto, la presente invención no debería limitarse a ninguna realización individual, sino que debería interpretarse con amplitud y en el ámbito de acuerdo con las reivindicaciones adjuntas.

REIVINDICACIONES

1. Un sistema para enlazar comandos entre una fuente (206) que comprende una recopilación de estados de una aplicación subyacente y un destino (203) que es una interfaz (402) de usuario,
comprendiendo el sistema un motor (208) de enlace de datos que
recibe al menos una instrucción de enlace que asigna un comando a un elemento del destino y que comprende una ruta de enlace y una indicación de una fuente (332) de datos;
evalúa la al menos una instrucción de enlace y actualiza el elemento de destino a un valor asociado con el comando;
monitoriza una recopilación de objetos que comprende la fuente de datos para una notificación de cambio;
en respuesta a la detección de la notificación de cambio, consulta en una gráfica de objetos, que comprende al menos un primer objeto y un segundo objeto, para determinar dinámicamente qué objeto se representa por la ruta de enlace y para determinar un valor actualizado del comando; y
actualiza el destino asignado al comando al valor actualizado del comando, en el que el comando es un objeto de comando, y
en el que cada flecha que apunta de un objeto a otro en la gráfica representa una propiedad, en el que el primer objeto apunta al segundo objeto, y en el que el segundo objeto es el objeto de comando.

2. El sistema de la reivindicación 1, en el que el comando está asociado con un estado.

3. El sistema de la reivindicación 2, en el que el estado de comando se obtiene a partir de la fuente (206).

4. El sistema de la reivindicación 2, en el que el estado de comando está asociado con una capacidad de ser ejecutado o con una incapacidad de ser ejecutado.

5. El sistema de la reivindicación 1, en el que la al menos una instrucción de enlace comprende una instrucción en un lenguaje de marcado declarativo.

6. El sistema de la reivindicación 5, en el que el lenguaje de marcado declarativo comprende HTML, XML o XAML.

7. El sistema de la reivindicación 1, en el que el comando comprende un objeto asociado con un procedimiento ejecutable y un estado booleano asociado con una capacidad o incapacidad de un procedimiento de ejecución asociado con el objeto del comando a ejecutar.

8. Un procedimiento para enlazar un comando entre una fuente que comprende una recopilación de estados de una aplicación subyacente y un destino que es una interfaz (402) de usuario, comprendiendo el procedimiento:

recibir al menos una instrucción de enlace que define una asignación entre el comando y el destino y que comprende una ruta de enlace y una indicación de una fuente (332) de datos;
determinar un valor del comando;
actualizar el destino al valor del comando;
monitorizar una recopilación de objetos que comprende la fuente de datos para una notificación de cambio;
en respuesta a detectar la notificación de cambio, consultar en un gráfica de objetos, que comprende al menos un primer objeto y un segundo objeto, para determinar dinámicamente qué objeto se representa mediante la ruta de enlace y para determinar un valor actualizado del comando; y
actualizar el destino asignado al comando al valor actualizado del comando,
en el que el comando es un objeto de comando, y
en el que cada flecha que apunta de un objeto a otro en la gráfica representa una propiedad, en el que el primer objeto apunta al segundo objeto, y en el que el segundo objeto es el objeto de comando.

9. El procedimiento de la reivindicación 8, en el que, en respuesta a determinar que la al menos una instrucción de enlace no puede evaluar, el valor del comando se establece en nulo o en un valor predeterminado.

10. El procedimiento de la reivindicación 8, en el que, en respuesta a determinar que el valor del comando es nulo, el destino se deshabilita.

11. El procedimiento de la reivindicación 8, en el que el objeto de comando está asociado con el estado.

12. El procedimiento de la reivindicación 11, en el que el estado de comando se obtiene a partir de de una fuente de datos.

13. El procedimiento de la reivindicación 11, en el que el estado de comando está asociado con una capacidad de ser ejecutado.

14. El procedimiento de la reivindicación 8, en el que la al menos una instrucción de enlace comprende una instrucción declarativa en un lenguaje de marcado.

15. El procedimiento de la reivindicación 14, en el que el lenguaje de marcado es uno de HTML, XML y XAML.

16. Un medio legible por ordenador que comprende instrucciones ejecutables por ordenador para enlazar comandos

entre una fuente (206) que comprende una recopilación de estados de una aplicación subyacente y un destino (203) que es una interfaz (402) de usuario, que incluye:

recibir al menos una instrucción de enlace que define una asignación entre un comando de una fuente de datos y un elemento de una interfaz de usuario y que comprende una ruta de enlace y una indicación de la fuente (332) de datos;
determinar un valor para el comando:

actualizar el elemento de la interfaz de usuario al valor del comando;
monitorizar una recopilación de objetos que comprende la fuente de datos para una notificación de cambio;
detectar la notificación de cambio;
en respuesta a detectar la notificación de cambio, consultar en un gráfica de objetos, que comprende al menos un primer objeto y un segundo objeto, para determinar dinámicamente qué objeto se representa mediante la ruta de enlace y para determinar un valor actualizado del comando; y
actualizar el elemento de interfaz de usuario asociado con el comando al valor actualizado del comando,
en el que el comando es un objeto de comando,
en el que cada flecha que apunta de un objeto a otro en la gráfica representa una propiedad, en el que el primer objeto apunta al segundo objeto, y en el que el segundo objeto es el objeto de comando.

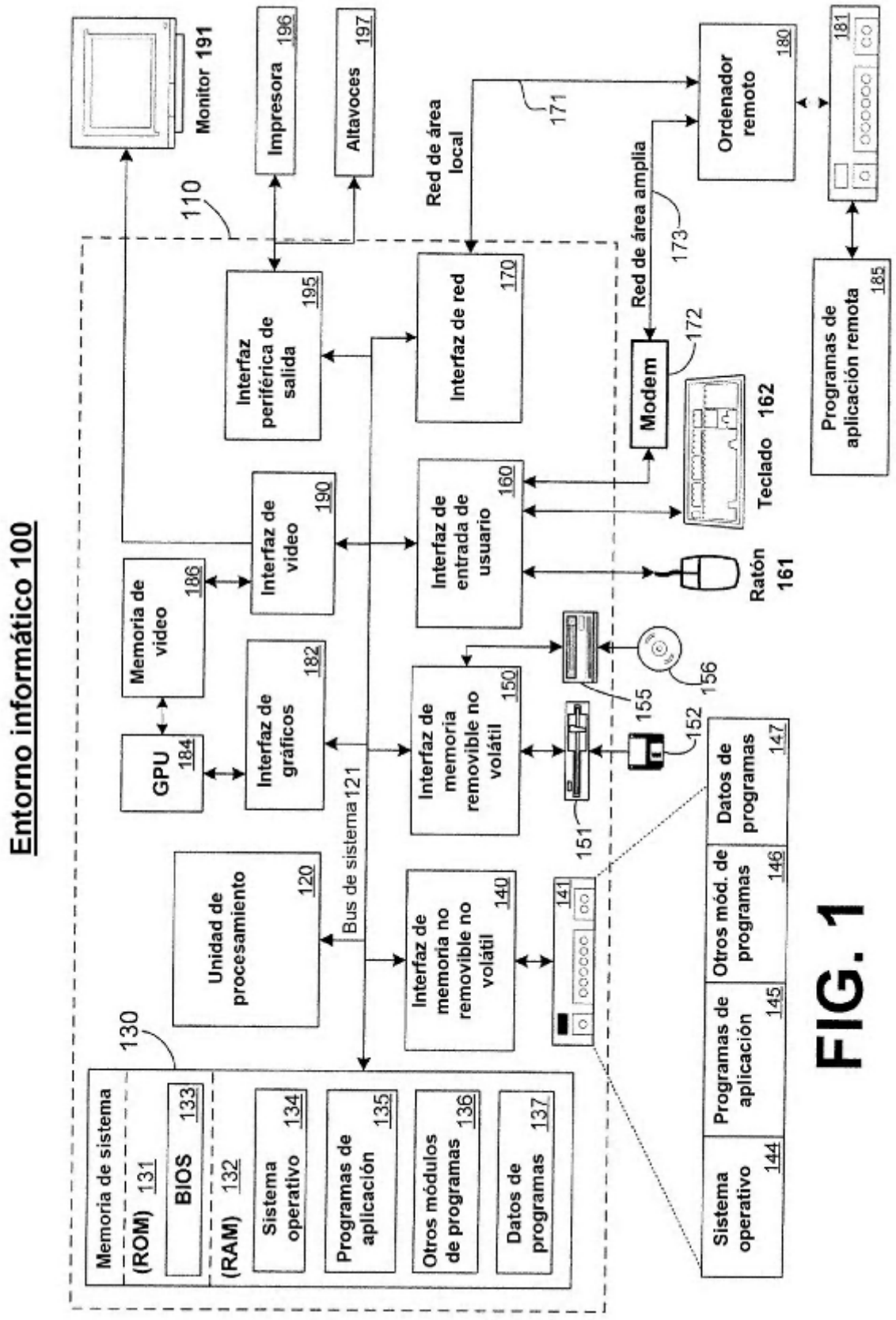
Entorno informático 100
Monitor 191
Impresora 196
Altavoces 197
110
Red de área local
171
Ordenador remoto 180
181
Red de área amplia
173
Interfaz periférica de salida 195
Interfaz de red 170
Modem 172
Programas de aplicación remota 185
Interfaz de video 190
Interfaz de entrada de usuario 160
Teclado 162
Ratón 161
Memoria de video 186
GPU 184
Interfaz de gráficos 182
Interfaz de memoria removible no volátil 150
155
156
152
151
Bus de sistema 121
Unidad de procesamiento 120
Interfaz de memoria no removible no volátil 140
141
Datos de programas 147
Otros mód. de programas 146
Programas de aplicación 145
Sistema operativo 144
130
Memoria de sistema (ROM) 131
BIOS 133
(RAM) 132
Sistema operativo 134
Programas de aplicación 135
Otros módulos de programas 136
Datos de programas 137

FIG. 1

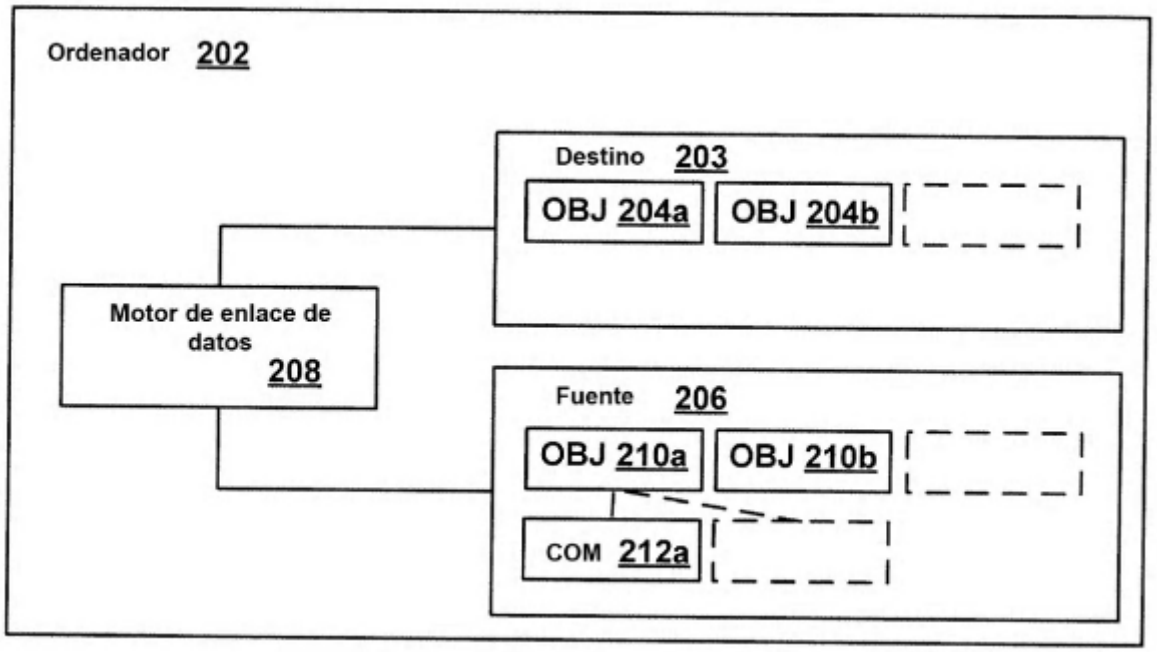
Ordenador 202
Destino 203
OBJ 204a
OBJ 204b
Motor de enlace de datos 208
Fuente 206
OBJ 210a
OBJ 210b
COM 212a

FIG. 2

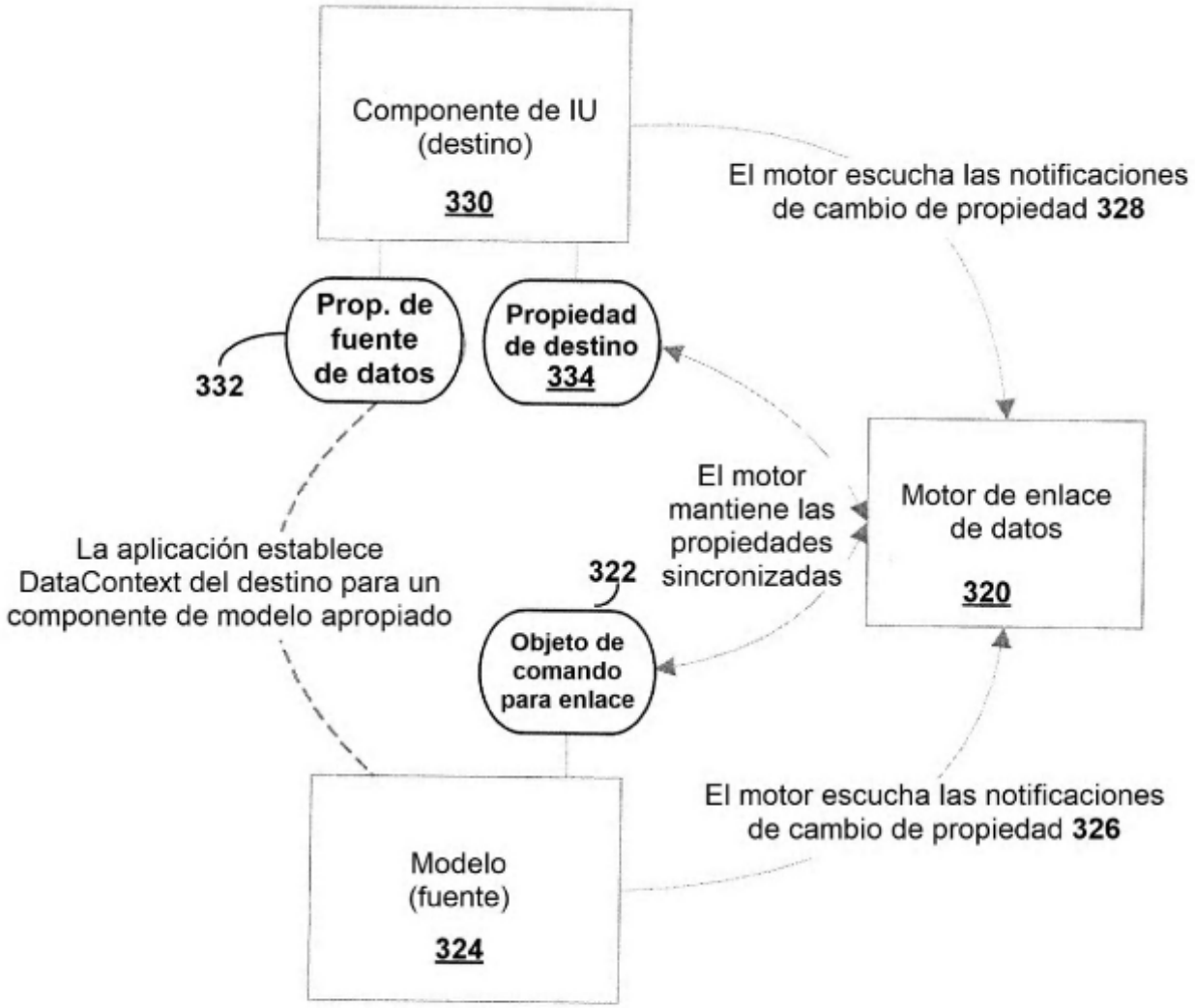
Componente de IU
(destino)
330
El motor escucha las notificaciones
de cambio de propiedad 328
Prop. de
fuente
de datos
332
Propiedad
de destino
334
El motor
mantiene las
propiedades
sincronizadas
Motor de enlace
de datos
320
La aplicación establece
DataContext del destino para un
componente de modelo apropiado
322
Objeto de
comando
para enlace
El motor escucha las notificaciones
de cambio de propiedad 326
Modelo
(fuente)
324

FIG. 3

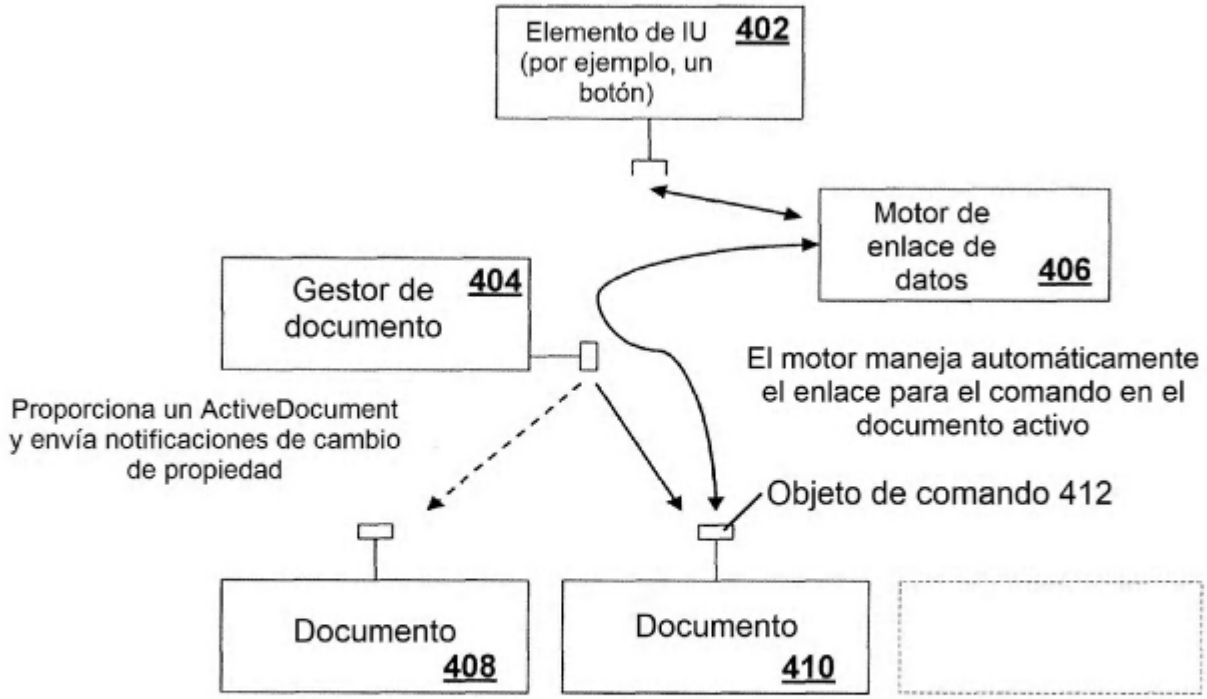
Elemento de IU (por ejemplo, un botón) 402
Motor de enlace de datos 406
Gestor de documento 404
El motor maneja automáticamente el enlace para el comando en el documento activo
Proporciona un ActiveDocument y envía notificaciones de cambio de propiedad
Objeto de comando 412
Documento 408
Documento 410

FIG. 4

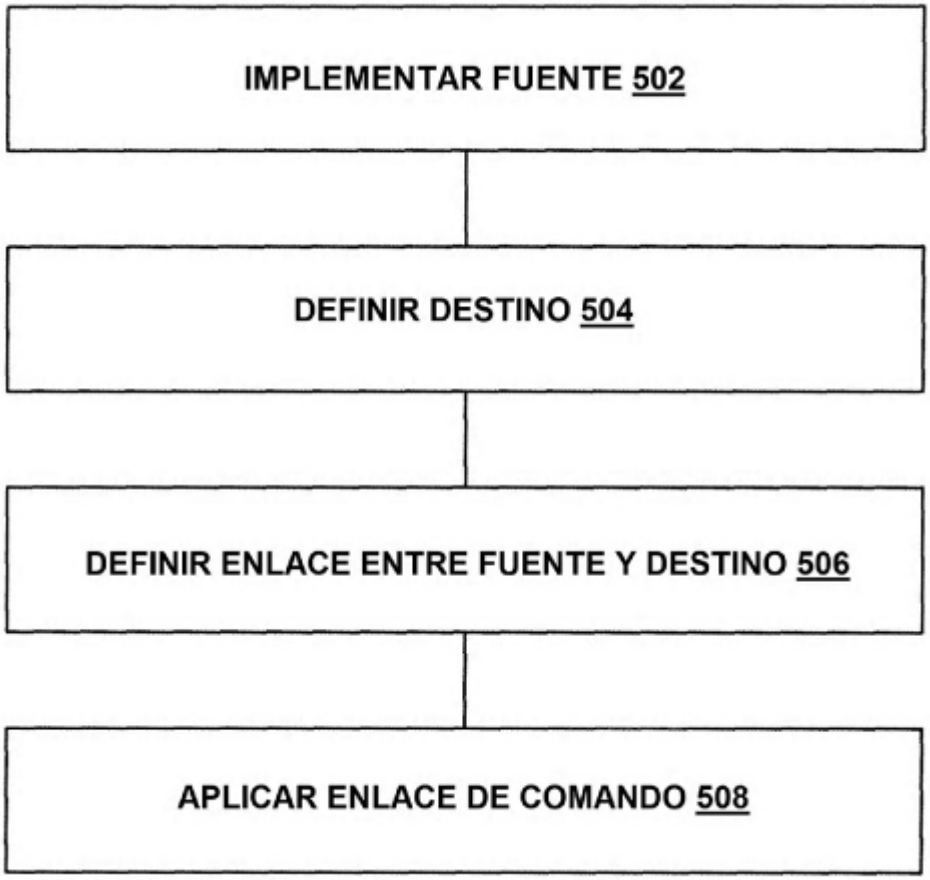
IMPLEMENTAR FUENTE 502
DEFINIR DESTINO 504
DEFINIR ENLACE ENTRE FUENTE Y DESTINO 506
APLICAR ENLACE DE COMANDO 508

FIG. 5

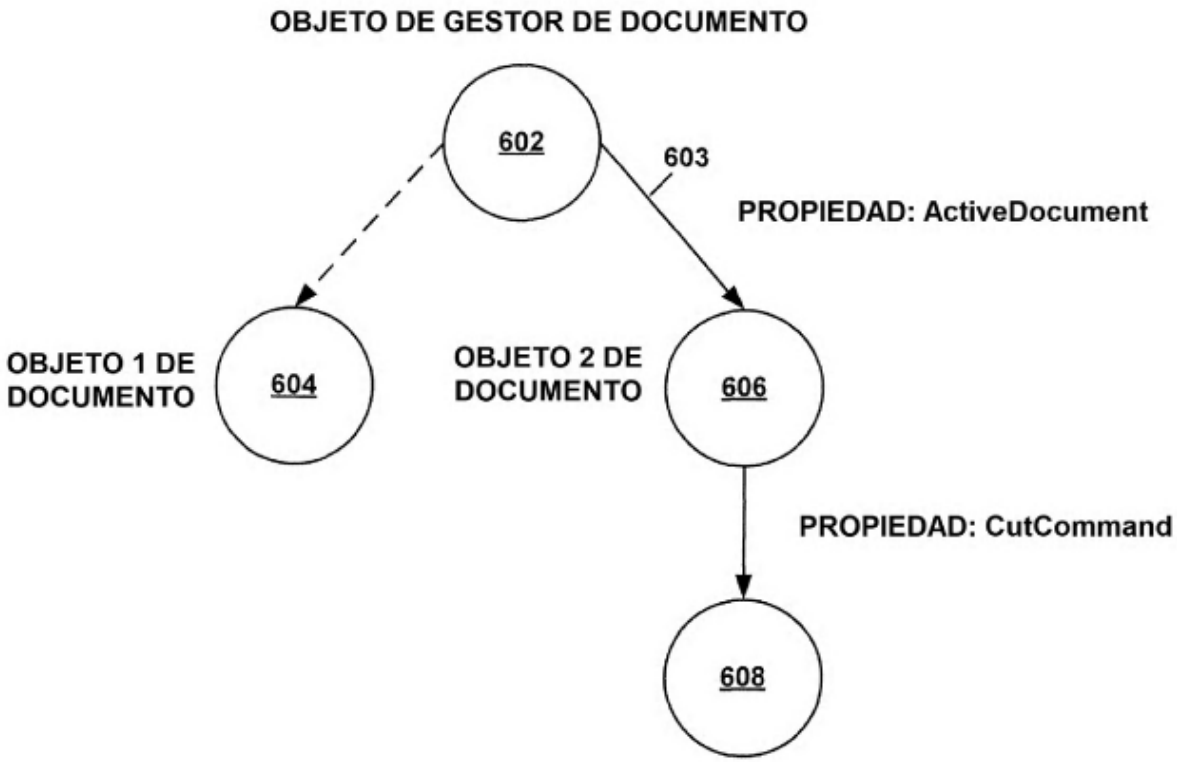
OBJETO DE GESTOR DE DOCUMENTO
602
603
PROPIEDAD: ActiveDocument
OBJETO 1 DE DOCUMENTO
604
OBJETO 2 DE DOCUMENTO
606
PROPIEDAD: CutCommand
608

FIG. 6