(12) SOLICITUD INTERNACIONAL PUBLICADA EN VIRTUD DEL TRATADO DE COOPERACIÓN EN MATERIA DE PATENTES (PCT)

(19) Organización Mundial de la Propiedad Intelectual
Oficina internacional

(43) Fecha de publicación internacional
17 de noviembre de 2016 (17.11.2016)

WIPO | PCT

(10) Número de Publicación Internacional
WO 2016/181368 A1

(51) Clasificación Internacional de Patentes:
G06F 9/45 (2006.01)

(21) Número de la solicitud internacional:
PCT/IB2016/052806

(22) Fecha de presentación internacional:
13 de mayo de 2016 (13.05.2016)

(25) Idioma de presentación: español

(26) Idioma de publicación: español

(30) Datos relativos a la prioridad:
62/161,216 13 de mayo de 2015 (13.05.2015) US

(72) Inventores; e
(71) Solicitantes : **HUEBRA, Nadia Analía** [AR/CO]; Calle 110 No. 7C - 46, Buenos Aires (AR). **HUEBRA, Mariano** [AR/AR]; Ruta Nacional 226 KM 163.4 S/N PUNTO TANDIL, Buenos Aires (CO).

(74) Mandatario: **OLARTE, Carlos R.**; Carrera 5 No. 34-03, La Merced, Bogotá, 110311 (CO).

(81) Estados designados *(a menos que se indique otra cosa, para toda clase de protección nacional admisible)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Estados designados *(a menos que se indique otra cosa, para toda clase de protección regional admisible)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), euroasiática (AM, AZ, BY, KG, KZ, RU, TJ, TM), europea (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declaraciónes según la Regla 4.17:

— *sobre la identidad del inventor (Regla 4.17(i))*

[Continúa en la página siguiente]

(54) Title: METHOD IMPLEMENTED BY A COMPUTER THAT PRESENTS SOFTWARE-TYPE APPLICATIONS BASED ON DESIGN SPECIFICATIONS

(54) Título : MÉTODO IMPLEMENTADO POR COMPUTADOR QUE EXPONE APLICACIONES TIPO SOFTWARE A PARTIR DE ESPECIFICACIONES DE DISEÑO

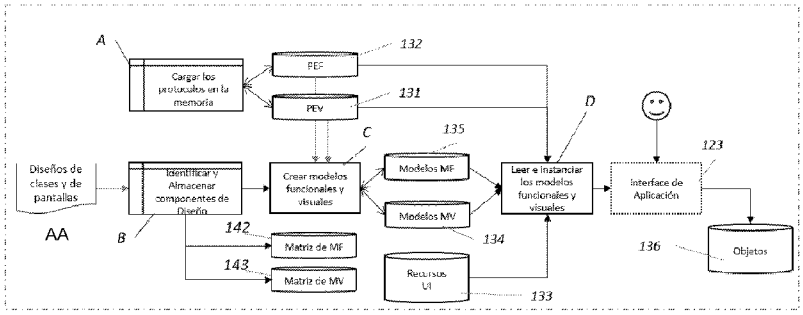


FIG. 2

AA	Designs of classes and displays
A	Loading the protocols into the memory
B	Identifying and storing design components
C	Create functional and visual models
D	Read and install the functional and visual models
123	Application interface
133	UI resources
134	MV models
135	MF models
136	Objects
142	MF matrix
143	MV matrix

(57) Abstract: The invention relates to a system and a method which, based on the input of logical information structures in an electronic device formed by a memory and a processor, automatically produces outputs in visual devices, which can be operated to create information relating to business procedures, replacing the software applications that are normally developed by means of the traditional programming known in the software industry. Given the information corresponding to the logical structure of a software design, this information is introduced into the machine, which stores same in a memory in the form of models. Said models are interpreted by the system (100) which automatically produces structures in visual devices, permitting the replacement of a software. The invention relates to the process used by the system (100) to introduce, store, interpret and produce the structural logic and visual structures that replace a software.

(57) Resumen:

[Continúa en la página siguiente]

WO 2016/181368 A1

Publicada:

— *con informe de búsqueda internacional (Art. 21(3))*

— *antes de la expiración del plazo para modificar las reivindicaciones y para ser republicada si se reciben modificaciones (Regla 48.2(h))*

El presente invento es un sistema y un método que, a partir de la entrada de estructuras lógicas de información en un dispositivo electrónico compuesto por memoria y procesador, produce de manera automática salidas en dispositivos visuales, que pueden ser operados para crear información de procesos de negocio, reemplazando a las aplicaciones de software que normalmente son desarrolladas mediante la programación tradicional conocida en la industria del software. Dada la información correspondiente a la estructura lógica de un diseño de software, esta información se ingresa en la máquina, que la almacena en una memoria en forma de modelos. Dichos modelos son interpretados por el sistema 100 que produce de manera automática estructuras en dispositivos visuales que permiten reemplazar a un software. El invento es el proceso que utiliza el sistema 100 para ingresar, almacenar, interpretar y producir las estructuras visuales y de lógica estructural que reemplazan a un software.

MÉTODO IMPLEMENTADO POR COMPUTADOR QUE EXPONE APLICACIONES TIPO SOFTWARE A PARTIR DE ESPECIFICACIONES DE DISEÑO

1. Referencia cruzada a solicitudes relacionadas

Esta solicitud reivindica los beneficios de la fecha de presentación de la solicitud de patente provisional U.S. N° 62/161,216 titulada "Método implementado por computador que expone aplicaciones tipo software a partir de especificación de diseño", que fue presentada el 13 de mayo de 2015, por los mismos inventores de esta solicitud. El contenido de dicha solicitud provisional se incorpora a la presente invención en su totalidad como referencia como si se expusiera en el presente documento.

2. Campo de la invención

El presente invento es un sistema y un método implementado por computador para producir automáticamente aplicaciones de tipo software completamente operables a partir de especificaciones de diseño de software.

El sistema permite ingresar diseños de clases y diseños de pantallas (especificaciones de diseño de software) a través de un dispositivo de entrada/salida. El procesador aplica dos protocolos para crear automáticamente modelos funcionales y visuales que se almacenan en una memoria de base de datos. Luego, combina estos modelos con componentes gráficos, e instancia automáticamente aplicaciones de tipo software, sin generación de código compilable, que se presentan al usuario para ser operadas y producir información como resultado.

3. Descripción del estado del arte

La industria del software propone como mecanismo de construcción de soluciones una serie de actividades entre las cuales se encuentran, la construcción de una base de datos y la codificación de un programa de computadora, que funcionan en el marco de una

arquitectura que permitirá al software resultante operar en entornos de múltiples usuarios.

Estas tareas son realizadas por personas capacitadas para comprender el diseño de un software y a partir de allí desarrollar el programa. La naturaleza de las tareas determina que la construcción de un software sea llevada a cabo por un equipo de trabajo donde se destacan los siguientes perfiles especializados:

- Arquitecto: define la arquitectura de capas de la aplicación teniendo en cuenta el entorno en el que operarán los usuarios. Ejemplos de arquitecturas son SOA (services oriented arquitectura), Cliente Servidor, etc.
- Programador de la base de datos: construye las estructuras donde serán almacenados los datos que el usuario genere con el uso de la aplicación. Existen herramientas llamadas motores de bases de datos de diferentes tecnologías pero que actualmente trabajan con estándares comunes ya establecidos.
- Programador de la aplicación: escribe el código de programa en un lenguaje definido (Java, C#, Vbnet, etc) que luego es compilado para obtener el código objeto que la computadora va a ejecutar, proporcionando la aplicación final de acceso al usuario.
- Constructor de la interface de usuario: construye pantallas, componentes visuales y plantillas de estilos gráficos que permiten al programa de computación presentarse ante el usuario de manera más amigable y usable.

Para llevar a cabo la construcción de un software, es necesario contar con un equipo profesional como el mencionado, ya que normalmente quien necesita hacer un software no tiene esta formación que demanda de varios años de estudio.

El problema que se presenta en la industria del software tiene varias aristas, a saber:

i. La demanda de soluciones crece a nivel mundial con una velocidad mayor a la velocidad de formación de profesionales para esta disciplina. Así se puede observar una gran demanda de desarrollo no abastecida.

ii. El proceso de construcción del software es costoso en tiempo y dinero, lo cual implica a los sectores demandantes no poder cubrir en su totalidad las necesidades de manera oportuna y conveniente.

iii. La multiplicidad de tecnologías (bases de datos, lenguajes de programación, arquitecturas, interfaces de usuario) que evolucionan a gran velocidad, generan en las aplicaciones ya desarrolladas un problema de obsolescencia tecnológica. Esto significa que cuando un software se encuentra operativo, normalmente aparecen nuevas tecnologías que los profesionales de la industria adoptan y la conversión de dicho software demanda un esfuerzo de re-ingeniería tan importante como su primera construcción.

iv. El cambio constante en las reglas de funcionamiento de los negocios en una economía global de interacción vertiginosa, demanda de adaptaciones y mejoras en las aplicaciones de software, que se vuelven inviables en muchas oportunidades debido la conjugación de los factores costo en tiempo/dinero y obsolescencia tecnológica. Por esta razón las aplicaciones de software no ofrecen evolución flexible y oportuna.

El arte previo divulga una pluralidad de sistemas y métodos para la generación automática de aplicaciones de software para reducir en tiempo y costos los ciclos de desarrollo de aplicaciones de software. El documento US2006/0015839 divulga un sistema y un método genera código fuente J2EE compilable, a partir de dos posibles fuentes, una base de datos en donde se especifican en detalle todos los parámetros, tablas y arquitectura del sistema a ser desarrollado, es decir que requiere como input el conocimiento detallado del diseño técnico. A partir de esta definición detallada construye un archivo XML que puede ser intervenido por el usuario para ajustar el diseño. El documento XML final es procesado por una herramienta marco (framework) que automáticamente genera el código J2EE y la arquitectura del sistema. Una herramienta de formas genera automáticamente las formas de interfaz de usuario, también basado en las estructuras definidas en el documento XML fuente. Una herramienta de despliegue (deployment) del sistema integra la base de datos fuente y el código del programa en una estructura compilable logrando así la aplicación de software resultante.

A diferencia del presente invento, el documento US2006/00115839 genera código J2EE compilable y requiere de un conocimiento previo de la arquitectura informática detallada del sistema a ser desarrollado y de la estructura de la base de datos, parámetros

objeto y de campo de tablas. También requiere de un conocimiento detallado de XML y en algunos casos de la introducción de código directamente, para ejecutar algunas acciones no realizadas por los programas de configuración y edición presentados.

Así mismo, el documento US2002/0077823 divulga un método y un sistema para crear aplicaciones de software que puedan operar en múltiples plataformas cliente, particularmente incluyendo plataformas interactuadas mediante voz, lo que implica un método para reconocer voz e interpretarlo como datos de entrada. Por ejemplo, un usuario de un dispositivo móvil con limitaciones de pantalla o teclado puede recibir información de una página web en formato de audio, y de forma recíproca puede interactuar con la página web utilizando palabras y frases habladas en lugar de escritas. Dicho documento se limita a analizar el lenguaje hablado para transformarlo en texto, el cual es posteriormente analizado como cualquier dato de entrada de una aplicación. A diferencia de la presente invención, dicho documento no pretende construir un modelo de software, ni sus documentos de arquitectura y especificación, automáticamente. Dicho documento se apoya en VoiceXML, el lenguaje estándar de especificación de componentes de voz. Para interpretar los comandos recibidos, utiliza plantillas de conversación de voz modificables manualmente, que son guiones manuales con los cuales el programador debe caracterizar el flujo de los comandos recibidos y las respuestas que debe brindar la aplicación. Para definir aquellos comandos o frases orales que la aplicación puede reconocer cuando son hablados por el usuario, el desarrollador debe crear una 'gramática' que es una caracterización explícita de los comandos hablados esperados y sirve para restringir las entradas a esos comandos, a un número pequeño de secuencias permitidas de palabras.

El sistema planteado en esta patente también incluye un mecanismo para que el programador defina "gramáticas en lenguaje natural", que constituyen una facilidad más para entrenar la aplicación con el tipo de respuestas esperadas, apoyado en plantillas, con sus propias frases. Luego es necesario que el programador manualmente vincule los componentes hablados identificados con variables de la aplicación para ser subsiguientemente procesados. Finalmente, para asegurar que dichos componentes hablados son correctamente identificados, es necesario contar con una base de datos de

patrones acústicos y pronunciaciones. Dicho documento proporciona un mecanismo para recibir comandos de voz y transformarlos en variables de entrada de una aplicación, y de una manera equivalente extrapolar el caso inverso para exponer variables de salida como comandos de voz autogenerados. Para ello, se apoya en la especificación de "gramáticas de lenguaje natural" y "modelos de lenguaje", pero, a diferencia del presente invento, no divulga la capacidad de analizar directamente el lenguaje natural de entrada, de manera libre y exhaustiva, para inferir automáticamente modelos de software, componentes funcionales o de diseño de una aplicación.

Así mismo, el documento US2012/0210296 divulga un método para crear aplicaciones de software tipo BPM (Business Process Management). Utiliza módulos de software pre-creados, que representan diferentes funcionalidades referidas a procesos y se ensamblan para lograr la aplicación de software resultante, a partir de la selección que un usuario hace, utilizando palabras del lenguaje natural predefinidas y asociadas a los módulos en una meta-data. El usuario puede escoger los procesos de una lista de términos que es expuesta y manejada por el sistema vinculada a los módulos existentes y disponibles. El usuario puede también implementar funciones que no estén en principio disponibles en la lista, pero que define como combinación de módulos existentes y se agregan a la lista de términos.

El presente invento, a diferencia de la patente US2012/0210296, no utiliza funcionalidades de procesos pre-creadas, permite el uso del lenguaje natural libre y a partir de él construye la funcionalidad creando modelos que son instanciables como aplicación de software, no solo para soluciones BPM, sino para todo tipo de solución que pueda describirse en lenguaje natural. El presente invento no implica un método de generación de código compilable, ni limita los resultados a una lista finita de funcionalidades. La patente US2012/0210296 no opera sobre la estructura del lenguaje, sino que utiliza el lenguaje para facilitar al usuario la selección de componentes técnicos de programas funcionales.

De acuerdo con lo anterior, y a pesar de que existe arte previo que automatiza actividades de desarrollo, aún existe la necesidad de la intervención de profesionales

expertos en el ciclo de desarrollo, en especial en la depuración de errores. De esta manera, el presente invento reemplaza el proceso de programación por una máquina que produce software de manera automática a partir de un diseño de clases y pantallas que representan un caso de aplicación tipo software a construir, dado como input del proceso. La presente invención resuelve las aristas centrales del problema de la industria de la siguiente manera:

- permite crear aplicaciones de software sin la necesidad de realizar las tareas tradicionales de la industria, es decir, la construcción de una base de datos, programación del código compilable, construcción de una arquitectura específica, que actualmente se realizan de manera manual o automatizada en aquellas herramientas que generan código de programa a partir de especificaciones. El presente invento produce el resultado de manera automática sin generar código de programa, a partir de los modelos funcionales y visuales almacenados en una memoria de base de datos, logrados aplicando los protocolos del presente invento a los diseños de clases y pantallas provistos como input por un usuario.
- reduce el alto costo en tiempo y dinero, obsolescencia tecnológica, desabastecimiento de la demanda, permite que el producto evolucione de manera flexible y oportuna.

En el estado del arte se reportan invenciones que son intérpretes. En la industria del software, los intérpretes se diferencian de los compiladores o de los ensambladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los intérpretes sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción como código. La presente invención es un instanciador de modelos porque no resuelve el software a partir de la compilación en tiempo real de cada una de las instrucciones de un programa. En la presente invención no existe la necesidad de programar las instrucciones del programa, ya que son reemplazadas por modelos almacenados en una base de datos que, al ser invocados por un motor instanciador de modelos, mediante la ejecución de múltiples instrucciones de programa compiladas como parte del motor, crean en memoria las

funcionalidades representadas por los modelos. El flujo del programa está bajo el control del usuario y la invocación recurrente de unos modelos con otros, que se materializa a partir de las instrucciones compiladas en el motor.

Dado que el motor instanciador de modelos del presente invento no es limitativo a la resolución de algún caso en particular, éste es capaz de instanciar a partir de cualquier conjunto de modelos referidos a cualquier funcionalidad, la modificación de dicha funcionalidad se realiza agregando o quitando modelos de la base datos, sin necesidad de modificar un programa compilado. Esto es más fácil y más rápido que crear nuevos programas para nuevas funcionalidades y no requiere de testing de programa. Al utilizar intérpretes o compiladores para resolver la funcionalidad se hace necesario realizar testing de programa con el impacto que esta actividad genera en las iteraciones hasta lograr un software de calidad.

4. Glosario

Con el ánimo de facilitar la interpretación de los conceptos divulgados en esta patente, a continuación, se lista una serie de expresiones que indicarán el alcance de algunos conceptos utilizados en la presente invención.

Usuario Final: Es el actor que utilizará finalmente la solución a través de la interfaz de usuario proporcionada.

Interfaz de usuario: La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo.

Aplicación de software: se compone de dos partes: el conjunto de modelos conceptuales que describen el negocio y determinan las reglas de comportamiento funcional de la solución, y el conjunto de modelos de visualización que harán posible que esta funcionalidad se materialice en una interfaz de usuario, con la que un usuario interactúe.

Programación: La programación informática, se deberá entender como el proceso de codificar, depurar y mantener el código fuente de programas computacionales. El código fuente es escrito en un lenguaje de programación. El propósito de la programación es crear programas que exhiban un comportamiento deseado. El proceso de escribir código requiere frecuentemente conocimientos en varias áreas distintas, además del dominio del lenguaje a utilizar, algoritmos especializados y lógica formal. Programar no involucra necesariamente otras tareas tales como el análisis y diseño de la aplicación (pero sí el diseño del código), aunque sí suelen estar fusionadas en el desarrollo de pequeñas aplicaciones.

Paradigma de Programación Orientado a Objetos (OO): Es el paradigma de programación actualmente más usado. El núcleo central de este paradigma es la unión de datos y procesamiento en una entidad llamada "objeto", relacionable a su vez con otras entidades "objeto". Tradicionalmente datos y procesamiento se han separado en áreas diferente del diseño y la implementación de software. Esto provocó que grandes desarrollos tuvieran problemas de fiabilidad, mantenimiento, adaptación a los cambios y escalabilidad. Con la orientación a objetos y características como el encapsulado, polimorfismo o la herencia se permitió un avance significativo en el desarrollo de software a cualquier escala de producción.

Metaclases: En el paradigma de OO una metaclase es una clase cuyas instancias son clases. Este concepto se aplica en el protocolo de especificación funcional de la presente invención con algunas particularidades.

Código de programa: es el conjunto de instrucciones y datos a ser procesados por un computador, escrito en un lenguaje que el computador puede interpretar y ejecutar. El código en computación puede ser binario o código fuente escrito en un lenguaje de nivel superior.

Código procedural: se llama así al código de programa generado en el estilo de programación estructurada. Este estilo está basado en estructurar el código de un

programa en componentes, que reciben el nombre de procedimientos, subrutinas o funciones.

Intérprete: En ciencias de la computación, intérprete es un programa informático capaz de analizar y ejecutar otros programas. Los intérpretes se diferencian de los compiladores o de los ensambladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los intérpretes sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.

Compilador: Un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación a lenguaje de máquina. Este proceso de traducción se conoce como compilación.

Sistema: en el presente invento se debe entender como un conjunto de elementos electrónicos u opto-electrónicos que interactúan entre sí para exponer en un dispositivo visual, aplicaciones tipo software a partir de especificaciones de diseño.

Componente de UI: son porciones de código (HTML u otro) que se utilizan para dar forma visual a la exposición de información en un dispositivo de tipo monitor o similar. A estos componentes se los conoce como sets de componentes de terceros y son utilizados en la industria del software como complementos en el desarrollo de aplicaciones de software.

5. Breve descripción del invento

El presente invento divulga un sistema y un método que a partir de la entrada de estructuras lógicas de información en un dispositivo electrónico compuesto por memoria y procesador, produce de manera automática salidas en dispositivos visuales, que pueden ser operados para crear información de procesos de negocio, reemplazando a las aplicaciones de software que normalmente son desarrolladas mediante la programación tradicional conocida en la industria del software.

El método implementado por computador interpreta y expone aplicaciones tipo software a partir de especificaciones de diseño que comprende las etapas de:

a. cargar mediante un Dispositivo de Entrada/Salida **120**, el protocolo de especificación funcional (PEF), el protocolo de especificación visual (PEV) y los recursos UI y almacenarlos en una memoria de base de datos **130**;

b. identificar, validar contra los protocolos de la etapa a y almacenar componentes de diseño funcional y componentes de diseño visual a partir de especificaciones de diseño de software en la memoria de base de datos **140**;

c. crear automáticamente modelos funcionales y modelos visuales a partir de los componentes de diseño funcional y visual identificados en la etapa b, y almacenar dichos modelos en la memoria de base de datos **130**, estructuras lógicas Modelos MF **135** y Modelos MV **134** respectivamente; y

d. leer, instanciar y exponer automáticamente, en el Dispositivo de Entrada/Salida **120**, Interface de Aplicación **123**, mediante el procesador **150**, configurado como instanciador de modelos **153**, los modelos funcionales y visuales creados en la etapa c, conjugados con los recursos UI, siguiendo las normas de los protocolos almacenados en la Etapa A.

Adicionalmente, el presente invento comprende un sistema que instancia y expone aplicaciones tipo software operativas a partir de especificaciones de diseño de software, que comprende:

a. un dispositivo de entrada/salida **120** configurado como Interface de CDF **121,** Interface de CDV **122**, Interface de Aplicación **123** para ingresar las especificaciones de diseño de software y exponer las aplicaciones tipo software resultantes;

b. una unidad de procesamiento central (CPU) **110** conectada al dispositivo de entrada/salida **120** que contiene:

- una memoria general **140** está en comunicación con el dispositivo de entrada/salida **120**, que interactúa con el procesador **150**, configurada para almacenar de manera volátil los componentes funcionales, visuales y componentes UI; y
- un procesador **150** configurado para recibir al menos una especificación de diseño de software a través del dispositivo de Entrada/Salida **120**; dicho procesador configurado como validador de modelos **151** para validar la especificación de diseño de software contra los protocolos de la Etapa A, e identificar los correspondientes modelos funcionales y visuales;

dicho procesador configurado como compaginador de UI **152** para compaginar la aplicación tipo software resultante, conjugando los modelos funcionales, visuales y recursos UI; y,

dicho procesador configurado como instanciador de modelos **153** para exponer la aplicación tipo software resultante en la Interface de Aplicación **123**; y

c. una memoria de base de datos **130** conectada a la CPU **110**, en comunicación con el procesador **150**, configurada para almacenar en forma estática, el protocolo de especificación Visual PEV en una estructura lógica PEV **131,** el protocolo de especificación funcional PEF en una estructura lógica PEF **132** y los recursos UI en una estructura lógica Recursos UI **133** y también configurada para almacenar en forma dinámica**,** modelos visuales en una estructura lógica Modelos MV **134**, modelos funcionales en una estructura lógica Modelos MF **135** y objetos en una estructura lógica Objetos **136**.

6. Breve descripción de las figuras

La FIG. 1. muestra la estructura del sistema del presente invento.

La FIG. 1A. muestra el protocolo de especificación funcional PEF.

La FIG. 1B. muestra el protocolo de especificación visual PEV.

La FIG. 1C. muestra la estructura lógica de la memoria de base de datos Modelos MF.

La FIG. 1D. muestra la estructura lógica de la memoria de base de datos Modelos MV.

La FIG. 1E. muestra la estructura lógica de la memoria de base de datos Objetos.

La FIG. 2. muestra las etapas del método del presente invento.

La FIG. 2A. muestra el detalle de subetapas de la Etapa A.

La FIG. 2B. muestra el detalle de subetapas de la Etapa B.

La FIG. 2B1. muestra un ejemplo de identificación de componentes funcionales y tipos OO.

La FIG. 2C. muestra el detalle de subetapas de la Etapa C.

La FIG. 2C1. muestra un ejemplo de aplicación del protocolo PEF sobre un diagrama de clases

La FIG. 2C2. muestra un ejemplo de aplicación del protocolo PEV sobre un diagrama de pantallas

La FIG. 2D. muestra el detalle de subetapas de la Etapa D

La FIG. 3A. muestra una modalidad adicional de las etapas del método del presente invento.

La FIG. 3B. muestra una modalidad adicional de las etapas del método del presente invento.

7. Descripción detallada de la invención

El presente invento corresponde a un sistema y método implementado por computador que permiten almacenar modelos de clases, que representan el diseño detallado de una aplicación de software en una memoria de base de datos, y exponen las aplicaciones de tipo software resultantes, a través de un proceso de instanciación de dichos modelos.

Como se describe en la Figura 2, el proceso comienza con el almacenamiento de los diseños de software en la memoria de base de datos, y luego identifica los componentes de dicho diseño, utilizando las definiciones de protocolos pre-cargados en el sistema para resolver la instanciación de los modelos, que finalmente constituyen la aplicación tipo software. En el presente invento, un modelo es la representación de un componente de diseño funcional o visual, materializada en una memoria de base de datos.

Para ejecutar el método, se utiliza un sistema como el descripto en la Figura 1, para ingresar los diseños de clases y pantallas en un Dispositivo de Entrada/Salida **120**. Los

diseños ingresados se transfieren a la memoria general para luego ser procesados por el Procesador **150**. El procesador se configura para validar, compaginar e instanciar los modelos existentes en las memorias de Base de Datos **134** y **135**. Los resultados de cada función de procesamiento se exponen a pantallas en un Dispositivo de Entrada/Salida **120.**

A continuación, se describen los componentes del sistema **100** de la Figura 1:

1. Dispositivo de Entrada/Salida **120**: es el dispositivo mediante el cual se ingresan los diseños de software. Permite a un procesador presentar en medios de visualización (pantallas, proyectores, televisores, impresoras, monitores, dispositivos móviles, entre otros) estructuras donde el usuario puede ingresar los diseños de clases y de pantallas (como los que se muestran en la Figura 2B), y visualizar las aplicaciones de tipo software resultantes, utilizando las siguientes configuraciones:
    a. Interface de CDF **121**: es una estructura visual que permite al usuario ingresar los componentes de diseño funcional de una aplicación de software a partir de un diagrama de clases, los cuales se almacenan en la Memoria de Base de Datos **130**, en la configuración lógica Modelos MF **135**.
    b. Interface de CDV **122**: es una estructura visual que permite al usuario cargar los componentes de diseño visual de una aplicación de software a partir de un diagrama de pantallas asociado a un diagrama de clases, que serán almacenados en la Memoria de Base de Datos **130**, en la configuración lógica Modelos MV **134**.
    c. Interface de Aplicación **123**: esta interface presenta al usuario las estructuras visuales de la aplicación de software resultante, una vez que el Procesador **150** instanció los modelos funcionales y visuales disponibles en la Memoria de Base de Datos **130**.
2. CPU **110**: es el dispositivo de procesamiento del sistema **100**. Este dispositivo está estructurado para realizar las funciones de validación de los modelos funcionales y visuales contra el respectivo protocolo, la compaginación de

recursos de UI y la instanciación de la aplicación de tipo software a partir de los pasos previos. Contiene la memoria general que permite el intercambio entre las funciones y el resto de los componentes del sistema.

a. Memoria General **140**: es un almacenamiento volátil que se utiliza como dispositivo de intercambio entre el dispositivo de entrada/salida, la memoria de base de datos y el procesador. Realiza las siguientes funciones según su configuración:
    i. Matriz de Componentes UI **141**: es la configuración de la Memoria General que posibilita el acceso a los recursos de UI existentes en la Memoria de Base de Datos **133** y los dispone para ser expuestos por el Procesador **150** como parte de la aplicación de tipo software resultante.
    ii. Matriz de MF **142**: es la configuración de la Memoria General que posibilita el procesamiento los Modelos Funcionales residentes en la Memoria de Base de Datos **135** mediante el Procesador **150** configurado como Instanciador de Modelos **153**, en correlación con los protocolos almacenados en las memorias de base de datos **131** y **132**.
    iii. Matriz de MV **143**: es la configuración de la Memoria General que posibilita el procesamiento los Modelos Visuales residentes en la Memoria de Base de Datos **134** mediante el Procesador **150** configurado como Instanciador de Modelos **153**, en correlación con los protocolos almacenados en las memorias de base de datos **131** y **132**.
b. Procesador **150**: es el dispositivo en el cual se llevan a cabo las tareas de procesamiento e intercambio. Realiza las siguientes funciones según su configuración:
    i. Validador de Modelos **151**: es la configuración del Procesador que se encarga principalmente de validar los modelos funcionales ingresados por el usuario contra los protocolos residentes en las memorias de base de datos **131** y **132**, para una vez validados

almacenarlos en las memorias de base de datos **134** y **135** respectivamente.

ii. Compaginador de UI **152**: es la configuración del Procesador que se encarga principalmente de compaginar los recursos UI residentes en la memoria de base de datos **133** con los modelos visuales almacenados en la memoria de base de datos **134**, previo al proceso de instanciación.

iii. Instanciador de Modelos **153**: es la configuración del Procesador que se encarga principalmente de instanciar los modelos funcionales y modelos visuales ya compaginados con los recursos de UI, residentes en las configuraciones lógicas de la memoria general **142** y **143**, para exponer automáticamente la aplicación tipo software resultante en la Interface de Aplicación **123**.

3. Memoria de Base de Datos **130**: es una memoria permanente que alberga los datos cargados por el usuario y generados por el Procesador **150** en sus diferentes configuraciones. Esta memoria contiene dos configuraciones de almacenamiento, una configuración estática y una configuración dinámica. La configuración estática almacena aquellos datos fijos necesarios que se cargan por única vez para el procesamiento que no son propios de "el caso". La configuración dinámica almacena los datos propios de "el caso" que se cargan por cada caso.

   a. Memoria Estática**:**

      i. PEF **131**: configuración de la Memoria de Base de Datos **130** que contiene las reglas que utiliza el Procesador **150** para validar e instanciar los modelos funcionales, definidas y cargadas en esta memoria a partir del Protocolo de Especificación Visual (PEV).

      ii. PEV **132**: configuración de la Memoria de Base de Datos **130** que contiene las reglas que utiliza el Procesador **150** para validar e instanciar los modelos funcionales, definidas y cargadas en esta memoria a partir del Protocolo de Especificación Funcional (PEF).

iii. Recursos UI **133**: configuración de la Memoria de Base de Datos **130** que contiene componentes de software que permiten al Procesador 150 exponer de los modelos visuales en la Interface de Aplicación **123**, mediante pantallas operables por el usuario.

b. Memoria Dinámica:

i. Modelos MV **134**: configuración de la Memoria de Base de Datos **130** que contiene los modelos funcionales cargados por el usuario mediante la Interface de CDV **122** y validados por el Procesador **150** contra el protocolo de Especificación Visual (PEV). Esta memoria, consta de las siguientes estructuras lógicas asociadas a los conceptos del Protocolo de Especificación Visual que se muestra en la Figura 1B:

1. Configuraciones asociadas al concepto ActionLink
    a. Estructura lógica RenderAction
    b. Estructura lógica RenderActionEditor
    c. Estructura lógica RenderAttributeBaseAction
    d. Estructura lógica RenderBaseViewAction
2. Configuraciones asociadas al concepto Application
    a. Estructura lógica Application
3. Configuraciones asociadas al concepto AttributeRenderInfo
    a. Estructura lógica RenderAttribute
    b. Estructura lógica RenderAttributeBase
    c. Estructura lógica RenderDiagramView
    d. Estructura lógica RenderMapView
    e. Estructura lógica RenderNavBarView
    f. Estructura lógica RenderObjectReferenceAttribute
    g. Estructura lógica RenderPivotView
    h. Estructura lógica RenderReportView
    i. Estructura lógica RenderWidgetAttribute
4. Configuraciones asociadas al concepto ColumnViewModel

a. Estructura lógica RenderColumn

5. Configuraciones asociadas al concepto Container
    a. Estructura lógica RenderContainer
6. Configuraciones asociadas al concepto ListViewModel
    a. Estructura lógica RenderFindView
    b. Estructura lógica RenderGridAggregateColumn
    c. Estructura lógica RenderGridAttribute
    d. Estructura lógica RenderGridColumn
    e. Estructura lógica RenderGridColumnAction
    f. Estructura lógica RenderGridView
7. Configuraciones asociadas al concepto Menu
    a. Estructura lógica Menu
8. Configuraciones asociadas al concepto MenuGroup
    a. Estructura lógica MenuGroup
9. Configuraciones asociadas al concepto TreeViewModel
    a. Estructura lógica RenderRelationView
    b. Estructura lógica RenderRelationViewAction
    c. Estructura lógica RenderTreeView
10. Configuraciones asociadas al concepto ViewModel
    a. Estructura lógica PopUpEditor
    b. Estructura lógica PrintConfiguration
    c. Estructura lógica RenderBaseView
    d. Estructura lógica RenderBaseViewInheritance

ii. Modelos MF **135**: configuración de la Memoria de Base de Datos **130** que contiene los modelos funcionales cargados por el usuario mediante la Interface de CDF **121** y validados por el Procesador **150** contra el protocolo de Especificación Funcional (PEF). Esta memoria, consta de las siguientes estructuras lógicas asociadas a los conceptos del Protocolo de Especificación Funcional, cuadrante More-Class, que se muestra en la Figura 1A:

1. Configuraciones asociadas al concepto Dominio
    a. Estructura lógica Domain

2. Configuraciones asociadas al concepto Formula
    a. Estructura lógica Fx
    b. Estructura lógica MatchAttribute
    c. Estructura lógica MatchFieldStrategy
    d. Estructura lógica MatchInfo
    e. Estructura lógica MatchValue
    f. Estructura lógica MatchValueStrategy
3. Configuraciones asociadas al concepto FxCall Argument
    a. Estructura lógica FxCall
4. Configuraciones asociadas al concepto ModelAttribute
    a. Estructura lógica MoreClassAttribute
5. Configuraciones asociadas al concepto ModelClass
    a. Estructura lógica MoreClass
    b. Estructura lógica MoreClassInheritance
6. Configuraciones asociadas al concepto ModelRelationClass
    a. Estructura lógica MoreRelationClass

iii. Objetos **136**: configuración de la Memoria de Base de Datos **130** que contiene los objetos que el usuario genera, basados en los modelos instanciados a través de la aplicación tipo software, por Procesador **150** configurado como Instanciador de Modelos **153**. Esta memoria, consta de las siguientes estructuras lógicas asociadas a los conceptos del Protocolo de Especificación Funcional, cuadrante Object-Class, que se muestra en la Figura 1A:

1. Configuraciones asociadas al concepto AttributeValue
    a. Estructura lógica MoreObjectAttributeValue
    b. Estructura lógica MoreObjectAttributeValueFileExtended
    c. Estructura lógica MoreRelationObjectMoreObjectAttributeRelation

2. Configuraciones asociadas al concepto Object
    a. Estructura lógica MoreObject
3. Configuraciones asociadas al concepto RelationObject
    a. Estructura lógica MoreRelationObject

La presente invención corresponde al sistema descripto y un método implementado en computador que interpreta y expone aplicaciones tipo software a partir de especificaciones de diseño. Haciendo referencia a la Figura 2, el método de la presente invención comprende las siguientes etapas:

Etapa A. Cargar a través del Dispositivo de Entrada/Salida **120** el Protocolo de especificación funcional (PEF) en la estructura lógica PEF **132 y** el Protocolo de especificación visual (PEV) con sus correspondientes recursos UI en las estructuras lógicas PEV **131** y Recursos UI **133** de la Memoria de Base de Datos **130**.

Etapa B. A partir de un diseño de clases, identificar y cargar los componentes de diseño funcional detallados en el diagrama de clases a través de la Interface CDF **121** y almacenarlos temporalmente en la Memoria General **140**, en la estructura lógica Matriz de MF **142.** A partir del diseño de pantallas, asociado al diseño de clases, identificar y cargar los componentes de diseño visual detallados a través de la Interface CDV **122** y almacenarlos temporalmente en la Memoria General **140**, en la estructura lógica Matriz de MV **143**.

Etapa C. Validar los modelos funcionales y visuales creados en la Etapa B, utilizando el Procesador **150** en su configuración de Validador de Modelos **151** para crear a partir de cada modelo cargado en la Memoria General **140**, un modelo compatible con los protocolos PEF y PEV residentes en la Memoria de Base de Datos **130** almacenados en la Etapa A. Almacenar los modelos funcionales y resultantes, en la Memoria de Base de Datos **130**, en sus configuraciones lógicas Modelos MF **134** y Modelos MV **133** respectivamente.

Etapa D. Recuperar de la Memoria de Base de Datos **130** los Modelos Funcionales y los Modelos Visuales creados en la Etapa C, los Recursos UI cargados en la Etapa A y almacenarlos en la Memoria General **140** utilizando el Procesador **150** configurado como Compaginador de UI **152**. Compaginar los Recursos UI con los componentes funcionales y visuales almacenados en la Memoria General **140** e instanciar los modelos compaginados y exponer las pantallas de una aplicación tipo software en el Dispositivo de Entrada/Salida **120**, Interface de Aplicación **123**, utilizando el Procesador **150** configurado como Instanciador de modelos **153.** Recibir a través de la Interface de Aplicación **123** las peticiones de los usuarios y almacenar los objetos resultantes de la respuesta del Procesador **150** a la operación de la aplicación, en la Memoria de Base de Datos **140**, estructura lógica Objetos **136**.

A continuación, se describen en detalle las etapas del proceso:

Etapa A. Cargar los protocolos en la memoria

En esta etapa, como se expone en la Figura 2A, se definen dos normas, llamadas protocolos, que se cargan en la Memoria de Base de Datos **130**, por única vez. El Procesador **150** utiliza estos protocolos, en sus diferentes configuraciones, a lo largo del proceso: i) Protocolo de especificación funcional (PEF); y ii) Protocolo de especificación visual (PEV).

Estos protocolos establecen el modo en que el Procesador **150** compaginará los componentes funcionales de un diagrama de diseño de clases de software (OO) y un diagrama de diseño de pantallas, para lograr una aplicación tipo software operativa de manera automática.

En el presente invento, un diagrama de clases con su respectivo diagrama de diseño de pantallas se denomina “el caso”. El diagrama de diseño de clases representa la funcionalidad de la aplicación de software (conformada por un conjunto de

componentes funcionales) y el diagrama de pantallas representa la estructura visual (conformada por un conjunto de componentes visuales) que presenta la aplicación tipo software a sus usuarios. A continuación, se describen ambos protocolos:

Subetapa A1. Cargar Protocolo de especificación funcional (PEF)

Este protocolo es la norma que define el comportamiento que el Procesador **150** dará a los componentes funcionales que el usuario cargue a partir de un diagrama de diseño de clases, de ahora en adelante componentes PEF, que determinan la estructuración lógica para la operación del Sistema **100**. La Arquitectura que define la funcionalidad de este Protocolo se puede observar en la Figura 1A.

El protocolo PEF define Metaclases que se refieren a clases cuyas instancias son clases, según OO. En este protocolo, las clases que son instancias de una Metaclase se conocen como Modelos (cuadrante Instance-Class de la Figura 1A).

Con el uso de este protocolo PEF, la funcionalidad de la aplicación tipo software resultante se logra instanciando el significado del lenguaje implícito en un diseño de clases, por un lado y la lógica matemática por otro lado. En la parte semántica de "el caso", los métodos (OO) identificados en el diagrama de clases, a través del protocolo se convierten en clases, y en la matemática los métodos (OO) que denotan una funcionalidad matemática se convierten en fórmulas. En el protocolo de Especificación Funcional PEF, como se observa en la Figura 1A cuadrante Model-Class, existen tres componentes para resolver la semántica (ModelClass, ModelRelationClass y ModelAttribute) y cuatro componentes para resolver la matemática (Dominio, Dominio externo, Fórmula y FxCall).

El protocolo también define clases de Objetos (cuadrante Object-Class de la Figura 1A). Estas clases de objetos ModelObject y ModelAttributeValue son los componentes que utiliza el sistema **100** para almacenar las instancias de los Modelos del cuadrante Instance-Object.

Los Componentes PEF son los pertenecientes a los cuadrantes Model-Class y Object-Class. Dentro del cuadrante Model-Class, se encuentran: ModelClass, ModelRelationClass, ModelAttribute, Fórmula, Dominio y FxCall. Son un conjunto de Metaclases que dan origen a los Modelos funcionales que se almacenan en la Memoria de Base de Datos **130**, en la estructura lógica Modelos MF **135**. Dentro del cuadrante Object-Class se encuentran: ModelObject y ModelAttributeValue. Son un conjunto de Clases que dan origen a los Objetos del Sistema que se almacenan en la Memoria de Base de Datos **130**, estructura lógica Objetos **136**. Estos componentes PEF se describen a continuación:

i. **Metaclases que dan Origen a los Modelos funcionales de la aplicación de software (cuadrante Model-Class)**

- ***ModelClass***

Es un componente metaclase porque sus instancias son modelos. Este componente modela estructuras de diseño que en el diagrama de diseño OO son clases. Un ModelClass puede heredar de otro ModelClass.

Ejemplo: Podemos crear una metaclase Entidad Humana (ModelClass), otra metaclase Persona (ModelClass que hereda del ModelClass Entidad Humana). Las instancias de Persona pueden ser Personas Físicas o Personas Jurídicas. Tanto Persona Física como Persona Jurídica son Modelos que constituyen instancias de la Metaclase Persona.

- ***ModelRelationClass***

Es un componente metaclase que hereda de ModelClass. La funcionalidad que extiende al ModelClass, agrega a esta metaclase la capacidad de relacionar dos ModelClass, que puede ser de dos tipos: asociación o composición, según se define en OO.

Ejemplo: Si tomamos el ModelClass Persona Física y el ModelClass Persona Jurídica, podríamos pensar que una instancia de Persona Física (Juan Pérez) puede ser empleado de una instancia de Persona Jurídica (Empresa Google). Este caso, existe un ModelRelationClass que se llama “se relaciona con” que vincula al ModelClass Persona consigo mismo a nivel de metamodelo. Esta relación podría utilizarse para representar la relación de empleado que existe entre persona Física y Persona Jurídica, o también podría especializar el ModelClass Persona en ModelClass Persona Física y Persona

Jurídica, para luego crear la especialidad de ModelRelationClass "es empleado de" que heredaría del ModelRelationClass "se relaciona con".

- ***ModelAttribute***

Es un componente metaclase que compone a ModelClass. Un ModelClass tiene una lista de ModelAttributes. ModelAttribute es la representación de los atributos que contendrán los modelos creados como instancias de ModelClass.

Un ModelAttribute define, entre otras cosas, el tipo de dato del atributo al cual está dando origen.

El protocolo define tres clasificaciones de tipos de dato para ModelAttribute.

- ModelAttribute Simple: los tipos simples son aquellos cuya instancia de valor no involucra instancias de ModelObject. Ejemplo: dato número, dato texto, dato imagen.
- ModelAttribute RelationObject: son atributos que existen en un ModelClass producto de que éste participe en un ModelRelationClass. Este tipo de atributo en su instancia de valor, dependiendo de la relación, puede tener una o más instancias de objetos de relación. Esto determina la cardinalidad de la relación. Ejemplo: Si tenemos un ModelRelationClass entre el ModelClass "Persona" y el ModelClass "Artículo", existe en el ModelClass Persona un ModelAttribute de tipo RelationObject que tendrá como instancia las Personas que se relacionen con Artículo. Y en la inversa, el ModelClass Persona tiene un ModelAttribute de tipo RelationObject que tendrá como instancias a los artículos que se relacionen con Persona.
- ModelAttribute ObjectReference: este tipo de datos es similar al anterior, solo que no surge producto de que el ModelClass al que pertenece participa en una relación. El atributo propiamente dicho es quien conoce la naturaleza de los objetos que oportunamente compondrán la instancia de valor para el mismo. Ejemplo: al crear un ModelAttribute como parte de un ModelClass lo declaro ObjectReference.

- ***Fórmula***

Una fórmula es un modelo independiente que cuenta con una **expresión** y un **conjunto de parámetros** para recibir información del Modelo que la invoca. La fórmula, cuando se ejecuta, siempre retorna un valor al Modelo que la llama. El valor resultante puede ser una única instancia o múltiples instancias como parte de arreglos y arreglos multidimensionales.

El protocolo de especificación funcional define al componente fórmula que se utiliza para resolver aquella lógica que funcionalmente requiere de la ejecución de operaciones matemáticas que utilizan como argumentos instancias de modelos. La fórmula provee todos los operadores matemáticos para operaciones algebraicas, matriciales o de conjuntos y permite definir la expresión haciendo uso de los argumentos existentes como instancias de modelos del cuadrante Model-Instance de la Figura 1A.

La funcionalidad que implica una resolución matemática se implementa mediante una Metaclase Fórmula. A diferencia del diseño tradicional de software, donde este tipo de funcionalidades se implementa como un método en una clase del modelo OO.

Ejemplo: una fórmula que calcula el valor total de una venta, cuenta con la expresión algebraica que suma los importes vendidos y devuelve un único valor resultante en una instancia de ModelAttribute, siendo los importes vendidos una colección de instancias de ModelAtrribute.

- ***Dominio***

El dominio es un componente que declara condiciones que permiten seleccionar un subconjunto de instancias de modelos (operaciones entre conjuntos).

Una instancia de modelo se puede encontrar en la memoria de Base de Datos **130**, en la estructura lógica Objetos **136**, ya validada y estructurada por el Procesador **150** en base al protocolo PEF, o en una memoria externa no estructurada en base al protocolo. En consecuencia, los Dominios se clasifican de la siguiente manera:

Dominios Internos

Son los dominios que se resuelven sobre la memoria de Base de Datos **130**, en la estructura lógica Objetos **136** obteniendo colecciones de instancias que luego podrán ser procesadas por otros modelos a través de fórmulas. Ejemplo: si existe un ModelClass Cliente, para recuperar sus instancias almacenadas en la base de datos se debe utilizar

un Dominio Interno, donde solo se invoca el nombre del ModelClass y la condición que deben cumplir las instancias a recuperar.

Dominios Externos

Son los dominios que se resuelven sobre una memoria NO estructurada en base al protocolo PEF (datos en archivos externos). En estos casos el componente dominio externo accede a las instancias de objetos y el Procesador **150** las estructura en base al protocolo para que formen parte del espacio accesible por dominios internos. Ejemplo: Si existe un archivo externo que tiene 3 columnas de datos: nombre, edad y empresa, para recuperarlos se debe crear un dominio externo donde se indica que dichos datos son datos de un cliente y la correspondencia de columnas con los ModelAttribute (columna Nombre: ModelAttribute Nombre, columna edad: ModelAttribute edad, columna empresa: ModelAttribute Empresa). El ModelAttribute empresa es de tipo RelationObject donde Cliente se relaciona con empresa.

- ***FxCall***

Este componente es el responsable de asociar cualquier instancia de ModelAttribute con las instancias de Fórmula, cuando el valor de la instancia de ModelAttribute proviene de una Fórmula. Es un modelo que resuelve la evaluación de una expresión de fórmula y retorna un resultado al modelo que la invoca, enviando al Procesador **150** los operadores matemáticos y los argumentos que serán operados. FxCall hidrata los valores de argumentos obtenidos de las instancias, para que la Fórmula invocada solicite resolución al Procesador **150**.

ii. Clases que dan Origen a los Objetos de la aplicación de software (cuadrante Object-Class)

- ***ModelObject***

Las instancias de las clases instanciadas a partir de ModelClass, se conocen como ModelObjects. ModelObject es la clase concreta que materializa las instancias de objetos modelados con ModelClass. Ejemplo: El modelo Artículo que es instancia de ModelClass posee una instancia concreta de ModelObject “leche”. Este objeto es a su vez la materialización de una instancia de ModelObject.

- ***ModelAttributeValue***

Las instancias de los modelos modelados con ModelAttribute se conocen como ModelAttributeValues. ModelAttributeValue es la clase concreta cuyas instancias representan el valor de los atributos de una instancia de ModelObject. Ejemplo: El

modelo Artículo que es instancia de ModelClass posee una instancia concreta de ModelObject "leche", cuyo ModelAttributeValue para el ModelAttribute nombre es "leche".

Subetapa A2. Cargar Protocolo de especificación visual (PEV)

Este protocolo define, como se muestra en la Figura 1B, el comportamiento que el Procesador **150** dará a los componentes visuales que el usuario cargue a partir de un diagrama de diseño de pantallas, de ahora en adelante componentes PEV. Estos componentes determinan la estructuración visual que se presentará en el Dispositivo de Entrada/Salida **120**, Interface de Aplicación **123** para la operación del Sistema **100**. La Arquitectura que define la presentación visual de este Protocolo se puede observar en la Figura 1B.

El protocolo de Especificación Visual PEV define el modo en que se vinculan los componentes PEF y los componentes PEV. Un componente PEV expone la funcionalidad de uno o más componentes PEF a través del Dispositivo de Entrada/Salida **120**, Interface de Aplicación **123**.

Esta definición del protocolo determina que es posible producir salidas en dispositivos visuales, que pueden ser operados para crear información de procesos de negocio sin hacer diseño de arquitectura de capas particular para cada caso, porque la arquitectura está embebida en los componentes PEF.

En otra modalidad del presente invento se crean componentes PEV no vinculados a componentes PEF, por ejemplo: un modelo visual que contiene un link a una página web.

Los Componentes PEV son: Application, ActionLink, AbmViewModel, ListViewModel, TreeViewModel, ReportViewModel.

- ***Application***

El componente Application estructura visualmente una aplicación de software.

El protocolo define el componente **Application** como un Modelo para especificar una aplicación. Una aplicación define como aspecto más importante el **Menu** mediante el cual el usuario tendrá acceso a la funcionalidad expuesta. Un **Menu** es una colección de

modelos **MenuGroup** y modelos **MenuItem** arborizada, donde los **MenuGroup** pueden contener a su vez otros **MenuGroup** o **MenuItem**. Los **MenuItem** son los que vinculan la acción que se llevará a cabo cuando el usuario haga click sobre el mismo. Un ítem tiene asociado alguno de los siguientes modelos de visualización (**ViewModel**): **AbmViewModel**, **ListViewModel**, **TreeViewModel**. Cuando el usuario ejecuta un **MenuItem**, el Procesador **150** del sistema **100** recibe dicha petición a través de la Interface de Aplicación **123** y devuelve una estructura en la misma Interface que permite al usuario ejecutar funcionalidad del componente PEF invocado por el componente PEV asociado al **MenuItem**.

- ***ViewModel***

El componente **ViewModel** es el que permite exponer instancias de ModelObject, en dispositivos visuales. Un componente ViewModel está compuesto por componentes **ActionLink** que permiten al **ViewModel** ejecutar acciones según el modelo ModelClass del cual es instancia y la clase de **ViewModel** a la que hace referencia.

Un componente PEV define un layout (estructura de disposición visual o plantilla) que será expuesta en el Dispositivo de Entrada/Salida **120**, Interface de Aplicación **123**, a partir de ViewModel y sus características.

Para exponer instancias de modelos, existen tres clases de ViewModel: AbmViewModel, ListViewModel y TreeViewModel con diferentes características y especificaciones de layout.

(1) AbmViewModel

El protocolo define el componente **AbmViewModel** como un Modelo para especificar el renderizado de una pantalla que permite dar de Alta, Baja o Modificar un Objeto ModelObject. El renderizado de una pantalla, expone todos o algunos de los ModelAttributeValues de una instancia de ModelObject, según el modelo ModelClass correspondiente.

El componente **AbmViewModel**, está compuesto por componentes **ActionLink** por ser un ViewModel y extiende su funcionalidad a través de los componentes **ControlViewType** y las características especiales del **AbmViewModel**.

Los componentes **ControlViewType** exponen los ModelAttributeValues según el tipo de dato del atributo. En una modalidad del invento se expone la siguiente lista de tipos de atributos:

- TextEditor: expone ModelAttributeValues de tipo texto.
- NumericEditor: expone ModelAttributeValues de tipo número.
- DateTimeEditor: expone ModelAttributeValues de tipo fecha.
- ObjectReferenceComboEditor: expone ModelAttributeValues de tipo selector de ModelObjects.
- ObjectReferenceListEditor: expone ModelAttributeValues de tipo selector de ModelObjects.

Los componentes **ActionLink** que componen al **AbmViewModel** son:

- Guardar: guarda la instancia de ModelObject en el repositorio Transaccional.
- Cancelar: cierra la estructura visual expuesta por el AbmViewModel y descarta los cambios realizados por el usuario en la instancia de ModelObject.
- Guardar y Nuevo: guarda la instancia de ModelObject y crea una nueva instancia para que el usuario complete los valores de sus atributos, exponiéndola en la estructura visual del AbmViewModel.
- Imprimir: Imprime los valores de los atributos de la instancia que se está modificando.
- En alguna modalidad de la invención se agregan acciones que invocan a otros ViewModel.

Características especiales de un AbmViewModel:

- AttributeRenderInfo: este modelo se utiliza para indicar las particularidades de renderizado de un control que representa el valor de determinado atributo de una instancia de ModelObject.
- ControlTemplate: selector de control para renderizar un ModelAttribute. Dependiendo del tipo de dato del atributo, éste podrá ser representado de diversas formas acordes con la naturaleza del tipo. ControlTemplate es el concepto que permite indicar con cuál de todas esas posibles opciones será efectivamente renderizado el atributo.
- Container: dentro de AbmViewModel existe una estructura de contenedores a los efectos de poder organizar los controles en dispositivo visual. Se conoce como controles a cada uno de los componentes que expone visualmente un atributo. Esta estructura de contenedores es arborizada, es decir, puede un contenedor incluir a otros.

- ColumnViewModel: dentro de los contenedores, los controles se organizan en columnas y éstas internamente tienen un orden para los controles que contienen.

(2) ListViewModel

El protocolo define el componente **ListViewModel** como un Modelo para especificar el renderizado de una pantalla que permite listar colecciones de instancias de ModelObject. El renderizado de esta pantalla, expone los ModelAttributeValues que dicho ModelObject posee según el modelo ModelClass del cual es instancia, para una colección de instancias del ModelObject que se muestran en una tabla donde cada ModelAttributeValue ocupa una celda, donde la fila corresponde a la instancia y la columna al ModelAttribute.

El componente **ListViewModel**, está compuesto por componentes **ActionLink** por ser un ViewModel y extiende su funcionalidad a través de los componentes **SearchViewModel**, **ColumnViewModel**, **ObjectReferenceListGridEditor** y las características especiales del **ListViewModel**.
Los componentes **ActionLink** que componen al **ListViewModel** son: Nuevo; Eliminar; Editar; Imprimir; Ejecutar Fórmula

Adicionalmente, para listas de RelationObjects surgen las siguientes acciones:

- NuevoMDestino: invoca la creación de un nuevo ModelObject del modelo relacionado como destino de la relación.
- NuevoMDestinoYAsociar: invoca la creación de un nuevo ModelObject del modelo relacionado como destino de la relación, y materializa la relación con el ModelObject propietario del atributo de relación.
- EliminarMDestino: en una instancia de relación, elimina el ModelObject relacionado, como así también la instancia de relación.
- EditarMdestino: invoca la pantalla de edición del ModelObject relacionado.

Características especiales de un ListViewModel:

- Herarchical view: se refiere a la representación con estructura jerárquica del grafo de objetos relacionados mediante modelos ModelRelationClass.

- InLine Edition: soporte para la modificación de los atributos de un objeto sobre la misma línea de la lista.

(3) TreeViewModel

El protocolo define el componente **TreeViewModel** como un Modelo para especificar el renderizado de un árbol de ModelObjects relacionados mediante RelationObjects. El renderizado de esta pantalla, consiste en la exposición de todos o algunos de los ModelAttributeValues que dicho RelationObjects invoca a partir de los ModelClass que relaciona.

Un nodo de un árbol, representado en un componente **TreeNodeModel,** representa entonces una instancia de ModelObject y una instancia de RelationObject, que es la relación que existe entre el nodo en cuestión y su nodo padre. Por lo tanto, los nodos del primer nivel del árbol no tendrán representación de ninguna instancia de RelationObject ya que no se relacionan con ningún nodo padre.

El componente **TreeViewModel**, está compuesto componentes **ActionLink** por ser un ViewModel y extiende su funcionalidad a través de las características especiales del **TreeViewModel**.

Los componentes **ActionLink** que componen al **TreeViewModel** son:

- Nuevo: crea una nueva instancia de RelationObject basada en el ModelRelationClass correspondiente según el modelo de visualización de árbol para el nodo donde se está invocando la acción.
- Eliminar: Elimina la instancia de RelationObject que vincula al nodo actual con su nodo padre.
- Editar: edita la instancia de RelationObject que vincula al nodo actual con su nodo padre.
- NuevoMDestino: invoca la creación de una nueva instancia de ModelObject que se corresponde con el destino de la relación para el modelo de ModelRelationClass correspondiente según el modelo de visualización de árbol para el nodo donde se está invocando la acción.

- NuevoMDestinoYAsociar: ídem NuevoMDestino, solo que luego crea la instancia de RelationObject para vincular el nuevo ModelObject con el ModelObject representado por el nodo sobre el que se invocó la acción.
- EliminarMDestino: elimina la instancia de ModelObject representada por el nodo actual.
- EditarMdestino: edita la instancia de ModelObject representada por el nodo actual.

Características especiales de un TreeViewModel:

- LoadOnDemand: indica si el árbol debe ir descubriendo y cargando sus nodos a medida que el usuario interactúa expandiendo el árbol.
- ShowExpanded: indica que, al mostrarse el árbol, éste debe visualizarse con todos sus nodos expandidos.
- EditOnClick: indica que debe invocarse la edición del objeto representado por un nodo cuando el usuario hace click sobre el mismo.

(4) ActionLink

Los componentes **ActionLink** definen las acciones que están disponibles en el ViewModel para ejecutar sobre las instancias de ModelObject.

Cada subclase de **ViewModel** posee una lista de ActionLinks distinta, en relación con la funcionalidad que expone en el dispositivo visual.

(5) ReportViewModel

Un ReportViewModel contiene la especificación necesaria para que el Procesador **150** configurado como Instanciador de Modelos **153**, pueda interactuar con cualquier servicio de entrega de reportes, como por ejemplo Reporting Services.

Esto es posible ya que el repositorio de instancias de ModelObject es compatible con los métodos de consulta q estos servicios implementan, de forma que haciendo uso de los componentes de visualización que acompañan a estas tecnologías, se pueden generar e integrar reportes de una forma simple y con todas las características de reporting necesarias.

Subetapa A3. Cargar recursos de UI

El Protocolo de Especificación Visual PEV está diseñado de forma que el Procesador **150** pueda compaginar los componentes visuales con recursos de UI. El objetivo de esta compaginación es que el Procesador pueda rápidamente poner estos componentes a disposición para el modelado de la interfaz de Usuario. Esto se logra mediante el concepto de ControlViewType del Protocolo PEV cargado en la Subetapa A2, el cual se vincula directamente con un recurso de UI y permite desde un ViewModel indicar con qué recurso UI será presentada determinada porción de la interfaz de usuario en el Dispositivo de Entrada/Salida **120**, Interface de Aplicación **123**.

Los Recursos UI son porciones parciales declaradas en algún lenguaje de UI (ejemplo, HTML5) y cuya apariencia puede ajustarse con hojas de estilo en cascada (por ejemplo, css). Estos recursos, se almacenan en la Memoria de Base de Datos **130**, estructura lógica Recursos UI **133**. Luego, cuando el Procesador **150**, configurado como Compaginador de UI **152** ejecuta sus funciones, vincula estos Recursos UI con una instancia de la estructura lógica ControlViewType, residente en la Memoria de Base de Datos **130**, estructura lógica Modelos MV **134**, dejando estos modelos conjugados disponibles para ser utilizados en la exposición de la aplicación tipo software.

Etapa B. Identificar y Almacenar componentes de Diseño

El diseño se describe porque resulta necesario como entrada al proceso abarcado por el invento.

Los documentos de diseño, como se muestran en la Figura 2B, se refieren particularmente a dos tipos de diagramas conocidos para definir la funcionalidad y las estructuras visuales que definen una aplicación de tipo software:

- los diagramas de clases (**220**) que representan el diseño funcional. Estos diseños serán utilizados para la creación de los modelos funcionales.
- los diagramas de pantallas (**221**) que representan el diseño visual requerido. Estos diseños serán utilizados para la creación de los modelos visuales.

En esta etapa se parte del diseño funcional y el diseño de pantallas, y se llevan a cabo las siguientes subetapas:

Subetapa B1. Identificar los componentes del diseño funcional

A partir de un diseño de clases (**220**) que representa la lógica que se desea resolver, listar las clases que figuran en el diagrama. A partir del diagrama de clases identificar sus componentes teniendo en cuenta que el tipo de componente se define en base al paradigma OO y hacerle corresponder un componente ModelClass, según establece el protocolo PEF cargado en la Subetapa A1. Identificar el nombre de cada componente identificado como se muestra en el Ejemplo de la Figura 2B1, y determinar el componente PEF que le corresponde, tal como se puede observar en el ejemplo de la Figura 2C1, así:

- Si el diagrama expone una clase, identificar un componente funcional tipo OO Clase y determinar que le corresponde un componente PEF ModelClass.
- Si el diagrama expone dentro de una clase un atributo, identificar un componente funcional tipo OO Atributo y determinar que le corresponde un componente PEF ModelAttribute.
- Si el diagrama expone una relación, identificar un componente funcional tipo OO, Relación y determinar que le corresponde un componente PEF ModelRelationClass.
- Si el diagrama expone un método dentro de una clase, identificar un componente funcional Tipo OO Método y determinar:
    - Si el método refiere a un cálculo matemático algebraico, que le corresponde un componente PEF Fórmula.
    - Si el método refiere a una función matemática equivalente a la lógica de conjuntos, que le corresponde un componente PEF Dominio, pudiendo ser externo en caso de referir a datos externos al sistema.
- Si el diagrama expone una herencia, identificar un componente funcional tipo OO, herencia y determinar que le corresponde una relación de herencia entre dos ModelClass, como lo define el protocolo PEF en el cuadrante Model-Class.

El usuario, a través del Dispositivo de Entrada/Salida **120**, Interface de CD **121**, carga los componentes PEF identificados y el Procesador **150**, configurado como Validador de Modelos **151,** los almacena en la Memoria General **140**, estructura lógica Matriz de MF **142**.

Subetapa B2. Identificar los componentes del diseño visual

Tomando como referencia el diseño de pantallas que representa el modo en que se desea ver e interactuar con la aplicación tipo software que construirá automáticamente el Procesador **150** configurado como Instanciador de Modelos **153**, se describen a continuación los pasos a ejecutar para determinar los modelos visuales que se deben crear para exponer la aplicación tipo software para "el caso".

A partir de la lista de componentes PEF, se aplica el protocolo de Especificación Visual cargado en la Etapa A, para identificar los conceptos de tipo ModelClass y crear los modelos visuales que define el protocolo PEV (Application, AbmViewModel, ListViewModel, TreeViewModel), así:

a. Por cada ModelClass, se determina que se deberán crear los siguientes ViewModel: AbmViewModel, ListViewModel.
b. Por cada ModelRelationClass, se determina que se deberán crear los siguientes ViewModel: ListViewModel, TreeViewModel
c. Para exponer la totalidad de los componentes PEF se determina que se deberá crear un Modelo Application, un modelo MenuViewModel,un modelo MenuGroup y un modelo ItemGroup por cada ViewModel creado en los pasos previos.

En una modalidad del presente invento el sistema **100** permite editar los modelos visuales creados para lograr la mayor similitud con los diseños de pantallas correspondientes a "el caso" disponibles como input del proceso.

El usuario, a través del Dispositivo de Entrada/Salida **120**, Interface de CDV **122**, carga los componentes PEV identificados y el Procesador **150**, configurado como Validador de Modelos **151** los almacena en la Memoria General **140**, estructura lógica Matriz de MV **143**.

Etapa B'. Generar CF y CD a partir de lenguaje natural.

En una modalidad de la invención, esta Etapa B' es sustitivva de la Etapa B, como se muestra en la Figura 3A.

En esta etapa, el usuario Genera CF y CD a partir de lenguaje natural, por ejemplo, utilizando como se describe en la solicitud de patente **No. 15/141,748**, titulada "PROCESO Y SISTEMA PARA GENERAR DOCUMENTOS DE ARQUITECTURA FUNCIONAL Y DOCUMENTOS DE ESPECIFICACIÓN DE ANÁLISIS Y DE DISEÑO DE SOFTWARE DE MANERA AUTOMÁTICA A PARTIR DE LENGUAJE NATURAL", incorporada como referencia en su totalidad.

En esta modalidad, se ejecutan las siguientes subetapas:

Subetapa B'1. Obtener componentes funcionales

El Procesador **150** se conecta a la Memoria de Base de Datos, estructura lógica **172** identificada en la solicitud de patente **No. 15/141,748,** obtiene los componentes funcionales de "el caso" y los almacena en la Matriz de MF **142**.

Subetapa B'2. Identificar componentes visuales

El Procesador **150**, recorre la lista de componentes funcionales obtenidos en la Subetapa B'1 y aplica el Protocolo de Especificación Visual PEV definido en la Subetapa A2. Obtiene de cada componente funcional uno o más componentes visuales, tal como se muestra en la Fig 2C2 y los almacena en la Matriz de MV **143**.

Etapa C. Crear modelos funcionales y visuales

En esta etapa, el Procesador **150** toma de la Memoria General **140**, estructura lógica Matriz de MF **142** los componentes PEF creados en la Subetapa B1 y ejecuta las siguientes subetapas:

Subetapa C1. Identificar los componentes PEF y crear los Modelos Funcionales

En esta subetapa, como se muestra en la Figura 2C, el Procesador **150** identifica los componentes PEF disponibles en la Memoria General **140**, creados en la Subetapa B1 y realiza la especificación de conceptos PEF según el protocolo cargado en la Subetapa A1. Luego crea los Modelos Funcionales, los cuales especifican la funcionalidad que utilizará el Procesador **150** para instanciar la aplicación tipo software para "el caso". El Procesador 150 lleva a cabo los siguientes pasos:

1. Recorre la lista para identificar qué Concepto PEF corresponde crear en base al Protocolo de Especificación para cada Tipo OO de los componentes del diseño según se determinó en la Subetapa B1.

2. Para aquellos componentes del diseño cuyo Concepto PEF asociado es una fórmula, crea los siguientes componentes PEF adicionales:
    a. Crea un Componente PEF de tipo FxCall y lo vincula con el ModelAttribute que hace uso de la Fórmula dentro de la misma Clase.
    b. Si la lógica de la fórmula implica el uso de atributos que no pertenecen al componente de diseño dueño del método, crea un Concepto PEF de tipo Dominio para localizar los datos que actuarán como argumentos de la Fórmula.

El Procesador **150** almacena los Modelos Funcionales creados en la Memoria de Base de Datos **130**, estructura lógica Modelos MF **135**.

La conversión del diseño de clases input del proceso, al aplicar el protocolo de especificación funcional PEF, derivan en el conjunto de modelos que luego son interpretados por el Procesador **150**, configurado como Instanciador de Modelos **153**.

Subetapa C2. Identificar los componentes PEV y crear los Modelos Visuales

En esta subetapa, como se muestra en la Figura 2C2, el Procesador **150** identifica los componentes PEV disponibles en la Memoria General **140**, creados en la Subetapa B2 y realiza la especificación de conceptos PEV según el protocolo cargado en la Subetapa A2. Luego crea los Modelos Visuales, los cuales especifican la funcionalidad que utilizará el Procesador **150** para exponer la aplicación tipo software para "el caso" presentando estructuras de información en el Dispositivo de Entrada/Salida **120**, Interface de Aplicación **123**. El Procesador **150** lleva a cabo los siguientes pasos:

1. Recorre la lista de Concepto PEF creados en la Subetapa B2 en base a lo determinado en el protocolo de especificación visual (PEV).
2. Para cada Concepto PEV identificado en el paso anterior, crea los modelos Visuales según se establece en el protocolo PEV.
3. Crear un modelo visual de tipo Aplication con un nombre representativo de "el caso".
4. Crear un modelo visual de tipo Menú con un nombre representativo y un Grupo de Menú por cada uno de los componentes de diseño, asignando al grupo el nombre del componente de diseño.
5. Crear dentro de cada grupo de Menú, dos Menú ítem, uno para cada modelo visual creado por componente de diseño. Ejemplo: para el componente clase 1, crear un grupo de Menú llamado "Clase" y luego dos Menú Item: uno llamado "ABM Clase1" y otro llamado "List Clase 1". Asociar a cada Menu Item el modelo visual correspondiente con su nombre.

Los modelos visuales se editan sin modificar el vínculo entre el modelo visual y el modelo funcional. Al editar un modelo visual se cambian dimensiones y ubicaciones de los componentes en el layout del dispositivo de Entrada/Salida **120** sin alterar el vínculo que existe entre el Modelo visual y el modelo funcional respectivo, según lo establecen los protocolos cargados en la Etapa A.

El Procesador **150** almacena los modelos visuales en la Memoria de Base de Datos **130**, estructura lógica Modelos MV **134**.

Etapa C'. Incorporar modelos funcionales y visuales.

En una modalidad de la invención, esta Etapa C' es sustitutiva de la Etapa C, como se muestra en la Figura 3B.

En esta etapa, se incorporan modelos funcionales y visuales, creados por un usuario, aplicando manualmente los criterios de los Protocolos definidos en las Subetapas A1 y A2.

A través del dispositivo de Entrada/Salida **120**, un usuario carga en la Memoria de Base de Datos **130**, estructura lógica Modelos MF **135**, los modelos funcionales creados manualmente.

A través del dispositivo de Entrada/Salida **120**, un usuario carga en la Memoria de Base de Datos **130**, estructura lógica Modelos MV **134**, los modelos visuales creados manualmente.

Etapa D. Leer e instanciar los modelos funcionales y visuales como una aplicación de tipo software operable

En esta etapa, el Procesador **150**, configurado como Instanciador de Modelos **153** accede a la Memoria de Base de Datos **130** con el objetivo de construir modelos conjugados que aloja en la Memoria General **140**. Un modelo conjugado, es un modelo que combina un modelo Funcional MF (definido en base al protocolo PEF) con un modelo Visual MV (definido en base al protocolo PEV), por lo tanto, representa la funcionalidad y la definición de los modelos visuales asociados con algún Recurso UI **133**, conjugados en un único modelo que se presenta en la Interface de Aplicación **123**, como una aplicación tipo software accesible y operable por un usuario.

El resultado producido por el Procesador **150** configurado como Instanciador de Modelos **153**, se expone mediante un servicio de interpretación y entrega de aplicaciones de tipo software. Cuando un usuario se autentica en este servicio, lo primero que recibe es una interfaz de aplicación con la cual podrá comenzar a interactuar.

La interpretación, en el presente invento, es un proceso que se ejecuta como servicio y mediante la lectura de modelos conjugados (Modelos MF + Modelos MV + Recursos UI), resuelve frente a la interacción del usuario, las acciones que éste vaya ejecutando sobre una aplicación de tipo software expuesta en el Dispositivo de Entrada/Salida **120**, Interface de Aplicación **123**.

Como resultado principal de la ejecución del sistema **100** se obtiene:

- Una aplicación tipo software expuesta en la Interface de Aplicación **123**.
- Un set de información generada por el usuario, en interacción con la aplicación tipo software, comúnmente llamado "datos del usuario", que el Procesador **150** almacena en la Memoria de Base de Datos **130**, estructura lógica Objetos **136**.

Para lograr este resultado, el Procesador **150**, configurado como Instanciador de Modelos **153**, lleva a cabo las siguientes subetapas:

Subetapa D1. Leer e interpretar los modelos visuales MV

Los modelos (tanto funcionales como visuales) son estructuras que respetan los protocolos cargados en la Etapa A, almacenadas en la Memoria de Base de Datos **130**, estructuras lógicas PEF **132** y PEV **131** respectivamente.

En esta subetapa, el Procesador **150**, configurado como Instanciador de Modelos **153**, lee los modelos visuales de la Memoria de Base de Datos **130**, estructura lógica Modelos MV **134** que se encuentran almacenados y validados según las definiciones del protocolo PEV cargado en la Subetapa A2.

Una vez leído el modelo visual, el sistema **100** construye una estructura tipo layout que organiza la disposición de las instancias de ModelObject y ActionLinks sobre el layout y almacena dicha estructura en la Memoria General **140**, estructura lógica Matriz de Componentes UI **141**.

Subetapa D2. Leer e interpretar los modelos funcionales MF

En esta subetapa, el Procesador **150**, en su función de Instanciador de Modelos 153, lee los modelos funcionales de la Memoria de Base de Datos **130**, estructura lógica Modelos MF **135** que se encuentran almacenados y validados según las definiciones del protocolo PEF cargado en la Subetapa A1.

El Procesador **150** del sistema **100** procesa tres tipos de modelos funcionales: modelos de cálculo, modelos de datos persistidos y modelos de datos interactivos:

Modelos de cálculo: procesa las expresiones de las clases fórmula que se encuentran en el repositorio MF transportando la expresión al servicio que la resuelve devolviendo el resultado como una instancia de ModelAttribute a través de FxCall.

Modelos de datos persistidos: procesa la expresión de acceso a la Memoria de Base de Datos **130** que proveen las clases Dominio, las cuales actúan sobre el repositorio de Objetos **136**, ejecutando la que se encuentran en el repositorio MF.
Modelos de datos interactivos: procesa los datos que el usuario ingresa en la Interface de Aplicación **123**, de manera interactiva a través de los ActionLinks de cada ViewModel con el cual interactúa el usuario.

Una vez leído el modelo funcional, el Procesador **150** del sistema **100** interpreta e instancia una estructura tipo business y la almacena en la Memoria General **140**, estructura lógica Matriz de Componentes UI **141**, asociada a la estructura visual creada en la Subetapa D1, donde los vínculos entre los modelos visuales y modelos funcionales instanciados responden a la lógica de los protocolos cargados en la Etapa A.

Subetapa D3. Seleccionar y Compaginar los Recursos UI

Los Recursos UI son los componentes de software pre-alojados en la Memoria de Base de Datos **130**, estructura lógica Recursos UI, que al ejecutarse asociados a un modelo visual permiten la generación del dibujo de la pantalla en la Interface de Aplicación **123**. Son las definiciones de procesamiento de los modelos visuales. En la industria del Software se comercializan distintos sets de recursos de UI, que pueden ser incorporados al sistema como se indica en la Subetapa A3, para ofrecer diferentes modos de visualización para las aplicaciones tipo software resultantes.
Para completar cada modelo conjugado, el sistema **100** selecciona un componente de la Matriz MF **142**, su correspondiente componente de la Matriz MV **143** y un Recurso UI **133** compatible con el tipo de MV seleccionado. El Procesador **150**, configurado como Compaginador de UI **152**, compagina los componentes seleccionados entregando como argumento de las funciones del Recurso UI, las partes del modelo conjugado (Modelo MF + Modelo MV asociado) que el componente requiere para funcionar.

De esta manera el Procesador **150** expone un componente conjugado en el Dispositivo de Entrada/Salida **120**, repitiendo el procedimiento para cada uno de los componentes creados para “el caso”. Un componente expuesto constituye una pantalla operativa de una aplicación de tipo software que presenta botones de acción, casillas donde el

usuario puede ingresar datos y otras funcionalidades que estén abarcadas por el modelo conjugado. Por ejemplo: Un ModelClass Artículo que tiene un ModelAttribute nombre, se expone en un AbmViewModel que define una pantalla con un TextEditor para editar la instancia del MOdelAttribute Nombre. Esta pantalla define una función por cada ActionLink del Model View.

El ViewModel especifica que utilizará el Componente UI llamado "control TextBox 3d" para mostrar el nombre, por lo cual el Procesador 150 compagina el componente mencionado que se encuentra almacenado en la Memoria Rcursos UI 133 para completr el modelo conjugado. Esta compaginación permite que la casilla de texto nombre se muestre en el dispositivo de Entrada/Salida **120** con una apariencia 3d coloreada, según las características del componente "control TextBox 3d". El componente de tipo TextEditor, se conjuga con el Recurso UI textbox 3D y se asocia con la funcionalidad ModelAttribute para lograr que el Procesador **150** instancie en la Interface de Aplicación **123** un cuadro donde un usuario pueda ingresar un texto, que tenga una apariencia tridimensional y soporte la recepción del texto para luego persistir el dato en la Memoria de Base de Datos **130**, estructura lógica Objetos **136**.

Subetapa D4. Resolver las instancias de ModelClass invocadas

Una vez resuelta la compaginación en la Subetapa D3, se encuentran en la Memoria General 140, estructura lógica Matriz de Componentes UI **141**, los modelos conjugados que formarán la aplicación de tipo software resultante. El Procesador **150**, configurado como Instanciador de Modelos **153**, obtiene las instancias del ModelObject asociado al modelo conjugado (ModelClass-ViewModel-Recurso UI) que se expone al usuario. Los AttributeValues correspondientes a la instancia seleccionada se envían al dispositivo de Entrada/Salida **120**, Interface de Aplicación **123** según el layout definido en el modelo conjugado y de este modo se presenta ante el usuario una pantalla de aplicación tipo software operable.

Subetapa D5. Recibir y resolver peticiones de usuario

Una vez expuesta la visualización en el dispositivo de Entrada/Salida **120**, Interface de Aplicación **123**, el usuario acciona presionando algún botón, ingresando algún dato en alguna de las casillas, desplegando alguna lista de opciones, o accediendo a alguna de

las estructuras visuales disponibles. Cuando esto sucede, el Procesador **150**, configurado como Instanciador de Modelos **153**, ejecuta la función correspondiente al modelo funcional que conforma el modelo conjugado sobre el cual está accionando el usuario, a través de un ActionLink o cualquiera de las funcionalidades existentes, según se define en el protocolo PEF cargado en la Subetapa A1.

Subetapa D6. Exponer Instancias de ModelClass actualizadas

Una vez resuelto el modelo conjugado, el Procesador **150**, configurado como Instanciador de Modelos **153**, se produce una actualización de instancia del modelo conjugado que está expuesto en el dispositivo de Entrada/Salida **120**, Interface de Aplicación **123**. El Procesador **150** toma la instancia nueva y actualiza el dispositivo de Entrada/Salida **120** con los cambios producidos.

Ejemplo de Aplicación del Método:

Dado un caso, que consta de un diagrama de clases como el que se expone en la Figura 2C1 y un diagrama de pantallas como el que se expone en la Figura 2D, se ejecutan las etapas del método, así:

- ***Etapa A. Cargar los protocolos en la memoria***

Esta etapa es considerada la inicialización del sistema, dado que por única vez se cargan los protocolos y eventualmente se agregan Recursos UI al sistema.

- ***Etapa B. Identificar y Almacenar componentes de Diseño***

En esta etapa se toman el diseño de clases y el diseño de pantallas de "el caso" y se realizan las siguientes subetapas:

(i) Subetapa B1. Identificar los componentes del diseño funcional

Por cada uno de los componentes del diagrama de clases, se identifican los correspondientes componentes PEF, aplicando el protocolo de especificación funcional PEF, cargado en la Subetapa A1, así:

Tabla 1

Componente del diseño	TipoOO	Nombre	Concepto PEF	Nombre Modelo a crear
Clase 1	Clase	Clase 1	ModelClass	Modelo1

Clase 1	Atributo	Atributo1	ModelAttribute	Modelo Atributo1
Clase 1	Atributo	atributoClase2_Rel1	ModelAttribute	Modelo atributoClase2_Rel1
Clase 2	Clase	Clase 2	ModelClass	Modelo Clase 2
Clase 2	Atributo	Atributo6	ModelAttribute	Modelo Atributo6
Clase 2	Atributo	Atributo7	ModelAttribute	Modelo Atributo7
Clase 2	Atributo	atributoClase1_Rel1	ModelAttribute	Modelo atributoClase1_Rel1
Clase 3	Clase	Clase 3	ModelClass	Modelo 3
Clase 3	Atributo	Atributo1	ModelAttribute	Modelo Atributo1
Clase 3	Atributo	Atributo2	ModelAttribute	Modelo Atributo2
Clase 3	Método	Calcular	Formula	Calcular
Clase 3	Método	Validar(Datos_Validar)	Formula	Validar(Datos_Validar)
Clase 3	No existe	Datos_Validar	Dominio	Datos_Validar
Clase 3	No existe	FxCallCalcular	FxCall	FxCallCalcular
Clase 3	No existe	FxCallValidar	FxCall	FxCallValidar
Clase 4	Clase	Clase 4	ModelClass	Modelo 4
Clase 4	Atributo	Atributo3	ModelAttribute	Modelo Atributo3
Clase 4	Método	Buscar(Datos_Buscar)	Formula	Buscar(Datos_Buscar)
Clase 4	No existe	Datos_Buscar	Dominio	Datos_Buscar
Clase 4	No existe	FxCallBuscar	FxCall	FxCallBuscar
Clase 5	Clase	Clase 5	ModelClass	Modelo 5
Clase 5	Atributo	Atributo9	ModelAttribute	Atributo9
Clase 5	Método	Reportar	Formula	Reportar
Clase 5	No existe	Datos_Reportar	Dominio	Datos_Reportar
Clase 5	No existe	FxCallReportar:Datos_Reportar	FxCall	FxCallReportar:Datos_Reportar
Relacion1	Atributo	Atributo 8	ModelAttribute	Modelo Atributo 8
Relacion1	Atributo	RatributoClase1_Rel1	ModelAttribute	Modelo RatributoClase1_Rel1
Relacion1	Atributo	RatributoClase2_Rel1	ModelAttribute	Modelo RatributoClase2_Rel1
Relacion1	Relacion	Relacion1	ModelRelationClass	Modelo Relacion1

El usuario ingresa esta lista, columnas 1, 2 y 3 (Componentes del diseño, Tipo OO y Nombre) de la Tabla 1, a través del Dispositivo de Entrada/Salida **120**, interface de CDF

121 y el Procesador **150** la almacena en la Memoria General **140**, estructura lógica Matriz de MF **142**. Luego, el Procesador **150**, configurado como Validador de Modelos **151**, lee el protocolo de especificación funcional PEF cargado en la Subetapa A1, completa las columnas 3 y 4 (Concepto PEF y Nombre del modelo a crear) de la Tabla 1, y actualiza la Memoria General **140**, estructura lógica Matriz de MF **142** con estos datos.

(ii) Subetapa B2. Identificar los componentes del diseño visual

Por cada uno de los componentes del diseño de pantallas, se identifican los correspondientes componentes PEV, aplicando el protocolo de especificación visual PEV, cargado en la Subetapa A2, así:

Tabla 2

Componente funcional	Concepto PEF	Concepto PEV	Modelo Visual a crear
Clase 1	ModelClass	ABM View Model	ABM Clase 1
Clase 1	ModelAttribute	List View Model	List Clase 1
Clase 1	ModelAttribute		
Clase 2	ModelClass		
Clase 2	ModelAttribute	ABM View Model	ABM Clase 2
Clase 2	ModelAttribute	List View Model	List Clase 2
Clase 2	ModelAttribute		
Clase 3	ModelClass		
Clase 3	ModelAttribute		
Clase 3	ModelAttribute		
Clase 3	Formula	ABM View Model	ABM Clase 3
Clase 3	Formula	List View Model	List Clase 3
Clase 3	Dominio		
Clase 3	FxCall		
Clase 3	FxCall		
Clase 4	ModelClass		
Clase 4	ModelAttribute	ABM View Model	ABM Clase 4
Clase 4	Formula	List View Model	List Clase 4
Clase 4	Dominio		
Clase 4	FxCall		
Clase 5	ModelClass	ABM View Model	ABM Clase 5

Clase 5	ModelAttribute	List View Model	List Clase 5
Clase 5	Formula		
Clase 5	Dominio		
Clase 5	FxCall		
Relacion1	ModelAttribute		
Relacion1	ModelAttribute	List View Model	List View Relacion 1
Relacion1	ModelAttribute	Tree View Model	Tree View Relacion1
Relacion1	ModelRelationClass		

El Procesador **150**, configurado como Validador de Modelos **151**, toma la lisa de componentes funcionales residente en la Memoria General **140**, estructura lógica Matriz de MF **142** y la expone en el dispositivo de Entrada/Salida 120, interface de CDV **122,** tal como se muestra en la columna 1 y 2 (Componente funcional, Concepto PEF) de la Tabla 2. El Procesador **150**, lee el protocolo de especificación visual PEV cargado en la Subetapa A2, y completa las columnas 3 y 4 (Concepto PEF, Modelo visual a crear) y actualiza la Memoria General **140**, estructura lógica Matriz de MV **143** con estos datos.

- ***Etapa C. Crear modelos funcionales y visuales***

Para los componentes funcionales y visuales identificados en la Etapa B del presente ejemplo, se realizan las siguientes subetapas:

(iii) Subetapa C1. Identificar los componentes PEF y crear los Modelos Funcionales

El Procesador **150** identifica los componentes PEF disponibles en la Memoria General **140**, Matriz de MF **142**, correspondientes a las columnas 3 y 4 (Concepto PEF y Nombre del modelo a crear) de la Tabla 1, y realiza la especificación de conceptos PEF según el protocolo cargado en la Subetapa A1. Recorre la lista, identifica el concepto PEF, y crea los componentes definidos para la matemática en los métodos que así lo requieren según la definición del protocolo PEF. Esta especificación implica identificar el concepto PEF y sus correspondientes estructuras lógicas en la Memoria de Base de Datos **130**, para crear los modelos funcionales y almacenarlos en la estructura lógica Modelos MF **135**.

En el presente ejemplo, se obtiene como resultado de esta etapa la siguiente lista de modelos funcionales, almacenados en la Memoria de Base de Datos **130**, estructura lógica Modelos MF **135**:

Tabla 3

Concepto PEF	Nombre Modelo MF
ModelClass	Modelo1
ModelAttribute	Modelo Atributo1
ModelAttribute	Modelo atributoClase2_Rel1
ModelClass	Modelo Clase 2
ModelAttribute	Modelo Atributo6
ModelAttribute	Modelo Atributo7
ModelAttribute	Modelo atributoClase1_Rel1
ModelClass	Modelo 3
ModelAttribute	Modelo Atributo1
ModelAttribute	Modelo Atributo2
Formula	Calcular
Formula	Validar(Datos_Validar)
Dominio	Datos_Validar
FxCall	FxCallCalcular
FxCall	FxCallValidar
ModelClass	Modelo 4
ModelAttribute	Modelo Atributo3
Formula	Buscar(Datos_Buscar)
Dominio	Datos_Buscar
FxCall	FxCallBuscar
ModelClass	Modelo 5
ModelAttribute	Atributo9
Formula	Reportar
Dominio	Datos_Reportar
FxCall	FxCallReportar:Datos_Reportar
ModelAttribute	Modelo Atributo 8
ModelAttribute	Modelo RatributoClase1_Rel1
ModelAttribute	Modelo RatributoClase2_Rel1
ModelRelationClass	Modelo Relacion1

La estructura técnica de creación de los modelos funcionales se puede observar en la Figura 2C1, donde cada componente de la Tabla 3, se exhibe, relacionado con el concepto del Protocolo PEF que le corresponde.

(iv) Subetapa C2. Identificar los componentes PEV y crear los Modelos Visuales

El Procesador **150** identifica los componentes PEV disponibles en la Memoria General **140**, Matriz de MV **143**, correspondientes a las columnas columnas 3 y 4 (Concepto PEF, Modelo visual a crear) de la Tabla 2, y realiza la especificación de conceptos PEV según el protocolo cargado en la Subetapa A2.

En el presente ejemplo, se obtiene como resultado de esta etapa la siguiente lista de modelos visuales, almacenados en la Memoria de Base de Datos **130**, estructura lógica Modelos MV **134**:

Tabla 4

Componente funcional	Concepto PEV	Modelo Visual a crear
Clase 1	ABM View Model	ABM Clase 1
Clase 1	List View Model	List Clase 1
Clase 1		
Clase 2	ABM View Model	ABM Clase 2
Clase 2	List View Model	List Clase 2
Clase 2		
Clase 2		
Clase 3	ABM View Model	ABM Clase 3
Clase 3	List View Model	List Clase 3
Clase 3		
Clase 3		
Clase 3		
Clase 3		
Clase 3		
Clase 3		
Clase 4	ABM View Model	ABM Clase 4
Clase 4	List View Model	List Clase 4
Clase 4		
Clase 4		
Clase 4		
Clase 5	ABM View Model	ABM Clase 5
Clase 5	List View Model	List Clase 5
Clase 5		
Clase 5		

Clase 5		
Relacion1	List View Model	List View Relacion 1
Relacion1	Tree View Model	Tree View Relacion1
Relacion1		
Relacion1		

- ***Etapa D. Leer e instanciar los modelos funcionales y visuales como una aplicación de tipo software operable***

Subetapa D1. Leer e interpretar los modelos visuales MV

Se ejecuta el proceso indicado en la Subetapa D1 del invento, partiendo de la lista del ejemplo de la Tabla 4.

Subetapa D2. Leer e interpretar los modelos funcionales MF

Se ejecuta el proceso indicado en la Subetapa D2 del invento, partiendo de la lista del ejemplo de la Tabla 3.

Subetapa D3. Seleccionar y Compaginar los Recursos UI

Se compaginan los recursos UI seleccionados para mostrar las pantallas requeridas en el diseño de pantallas, hasta lograr un diseño visual como el de la Figura 2D asociada.

Subetapa D4. Resolver las instancias de ModelClass invocadas

A medida que el usuario opera sobre la Interface de Aplicación, es sistema resuelve, ejecutando las acciones enumeradas en la Subetapa D4 del presente invento.

Las siguientes subetapas se producen en la medida que el usuario interactúa con la aplicación resultante del ejemplo:

Subetapa D5. Recibir y resolver peticiones de usuario

Subetapa D6. Exponer Instancias de ModelClass actualizadas

Se debe entender que la presente invención no se halla limitada a las modalidades descritas e ilustradas, pues como será evidente para una persona versada en el arte, existen variaciones y modificaciones posibles que no se apartan del espíritu de la invención, el cual solo se encuentra definido por las siguientes reivindicaciones.

REIVINDICACIONES

1. Un método implementado por computador que instancia automáticamente y expone aplicaciones tipo software operativas a partir de especificaciones de diseño de software, que comprende las etapas:

   a. cargar mediante un Dispositivo de Entrada/Salida **120**, el protocolo de especificación funcional (PEF), el protocolo de especificación visual (PEV) y los recursos UI y almacenarlos en una memoria de base de datos **130**;

   b. identificar, validar contra los protocolos de la etapa a y almacenar componentes de diseño funcional y componentes de diseño visual a partir de especificaciones de diseño de software en la memoria de base de datos **140**;

   c. crear automáticamente modelos funcionales y modelos visuales a partir de los componentes de diseño funcional y visual identificados en la etapa b, y almacenar dichos modelos en la memoria de base de datos **130**, estructuras lógicas Modelos MF **135** y Modelos MV **134** respectivamente; y

   d. leer, instanciar y exponer automáticamente, en el Dispositivo de Entrada/Salida **120**, Interface de Aplicación **123**, mediante el procesador **150**, configurado como instanciador de modelos **153**, los modelos funcionales y visuales creados en la etapa c, conjugados con los recursos UI, siguiendo las normas de los protocolos almacenados en la Etapa A.

2. El método de la reivindicación 1 donde la etapa b comprende las subetapas:

   a. identificar los componentes del diseño funcional a partir de un diseño de clases cargados desde memorias o dispositivos de entrada; y

   b. identificar los componentes del diseño visual a partir del diseño de pantallas cargados desde memorias o dispositivos de entrada.

3. El método de la reivindicación 1 donde la etapa c comprende las subetapas:

a. procesar mediante una unidad de procesamiento central el protocolo de especificación funcional (PEF) para identificar sus componentes, y;

b. procesar mediante una unidad de procesamiento central el protocolo de especificación visual (PEV) para identificar sus componentes y crear los modelos visuales, dichos modelos visuales será almacenados en una memoria de bases de datos;

4. El método de la reivindicación 1 donde la etapa d comprende las subetapas:

a. leer y procesar mediante una unidad de procesamiento central los modelos visuales;

b. leer y procesar mediante una unidad de procesamiento los modelos funcionales;

c. realizar el procesamiento mediante una unidad de procesamiento central de los modelos funcionales para seleccionar y compaginar con los componentes de la interface de usuario;

d. procesar mediante una unidad de procesamiento central los componentes de los modelos funcionales para resolver las instancias de las metaclases invocadas por el usuario mediante su interacción a través de dispositivos de entrada y salida;

e. recibir y resolver las peticiones de usuario realizadas a través de dispositivos de entrada y salida y procesarlas por una unidad de procesamiento central; y

f. exponer en dispositivos de visualización las instancias de la metaclases actualizadas.

5. El método de la reivindicación 2 donde la subetapa a, comprende los pasos:

a. procesar mediante una unidad de procesamiento central los diagramas funcionales almacenados en la memoria para identificar las clases incluidas en dicho diagrama, y;

b. identificar el nombre de cada clase y almacenarlo en la memoria de base de datos.

6. El método de la reivindicación 2 donde la subetapa b comprende los pasos:

a. crear los modelos visuales a partir de las clases identificadas aplicando el protocolo de especificación visual y almacenarlo en la memoria de base de datos; y

b. editar la apariencia de los modelos visuales creados y almacenados en la memoria de base de datos utilizando dispositivos de entrada y salida en caso de ser requerido;

7. El método de la reivindicación 3 donde la subetapa a comprende los pasos:

a. procesar mediante una unidad de procesamiento central la lista de modelos funcionales para crear los objetos incluidos en dichos modelos funcionales siguiendo el protocolo de especificación funcional cargados en la memoria permanente para cada uno de los componentes; y

b. procesar mediante una unidad de procesamiento central los componentes de los diagramas funcionales que están asociados a una fórmula matemática para crear componentes adicionales que permitan la ejecución de dicha fórmula y almacenarlo en la memoria de base de datos.

8. El método de la reivindicación 3 donde la subetapa b comprende los pasos:

a. procesar a través de una unidad de procesamiento central la lista de conceptos funcionales almacenada en la memoria de base de datos para definir para cada concepto funcional un concepto del protocolo de especificación visual (PEV);

b. crear los modelos visuales para cada concepto funcional y almacenarlo en la memoria de base de datos;

c. crear un modelo visual para la aplicación con un nombre representativo del caso y almacenarlo en la memoria de base de datos; y

d. crear un modelo visual de tipo menú con un nombre representativo y un grupo de menú por cada uno de los componentes de diseño, asignando al grupo el nombre del componente de diseño y almacenarlo en la memoria de base de datos.

9. Un sistema que instancia y expone aplicaciones tipo software operativas a partir de especificaciones de diseño de software, que comprende:

a. un dispositivo de entrada/salida **120** configurado como Interface de CDF **121,** Interface de CDV **122**, Interface de Aplicación **123** para ingresar las especificaciones de diseño de software y exponer las aplicaciones tipo software resultantes;

b. una unidad de procesamiento central (CPU) **110** conectada al dispositivo de entrada/salida **120** que contiene:

. una memoria general **140** está en comunicación con el dispositivo de entrada/salida **120**, que interactúa con el procesador **150**, configurada para almacenar de manera volátil los componentes funcionales, visuales y componentes UI; y

. un procesador **150** configurado para recibir al menos una especificación de diseño de software a través del dispositivo de Entrada/Salida **120**; dicho procesador configurado como validador de modelos **151** para validar la especificación de diseño de software contra los protocolos de la Etapa A, e identificar los correspondientes modelos funcionales y visuales;

dicho procesador configurado como compaginador de UI **152** para compaginar la aplicación tipo software resultante, conjugando los modelos funcionales, visuales y recursos UI; y,

dicho procesador configurado como instanciador de modelos **153** para exponer la aplicación tipo software resultante en la Interface de Aplicación **123**; y

c. una memoria de base de datos **130** conectada a la CPU **110**, en comunicación con el procesador **150**, configurada para almacenar en forma estática, el protocolo de especificación Visual PEV en una estructura lógica PEV **131,** el protocolo de especificación funcional PEF en una estructura lógica PEF **132** y los recursos UI en una estructura lógica Recursos UI **133** y también configurada para almacenar en forma dinámica**,** modelos visuales en una estructura lógica Modelos MV **134**, modelos funcionales en una estructura lógica Modelos MF **135** y objetos en una estructura lógica Objetos **136**.

10. El sistema de la reivindicación 9 donde la unidad de procesamiento central se caracteriza por:

a. recuperar los modelos de diseño almacenados en el dispositivo de entrada y hacer el procesamiento de validación de dichos modelos de diseño contra los protocolos de validación funcional (PEF) cargados en la memoria permanente;

b. recuperar los modelos de diseño de interfaces de usuario almacenados en el dispositivo de entrada y hacer el procesamiento de validación de dichos modelos de diseño de interfaces de usuario contra los protocolos de validación visual (PEV) cargados en la memoria permanente;

c. realizar el procesamiento de interpretación de modelos funcionales y almacenarlos en la memoria de base de datos;

d. realizar el procesamiento de interpretación de modelos visuales y almacenarlos en la memoria de base de datos;

e. recuperar de la memoria de base de datos los modelos funcionales y los modelos visuales, realizar el procesamiento de compaginación e interacción entre ellos, crear la interfaces de usuario, almacenar dichas interfaces de usuario en la memoria permanente y presentar la aplicación tipo software en los dispositivos de entrada y salida

11. El sistema de la reivindicación 9 donde el dispositivo de entrada se caracteriza por:

a. permitir ingresar los diagramas funcionales a una memoria general que serán recuperados por la unidad de procesamiento central para crear los modelos funcionales;

b. permitir ingresar los diseños de pantallas a una memoria general que serán recuperados por la unidad de procesamiento central para crear los modelos visuales.

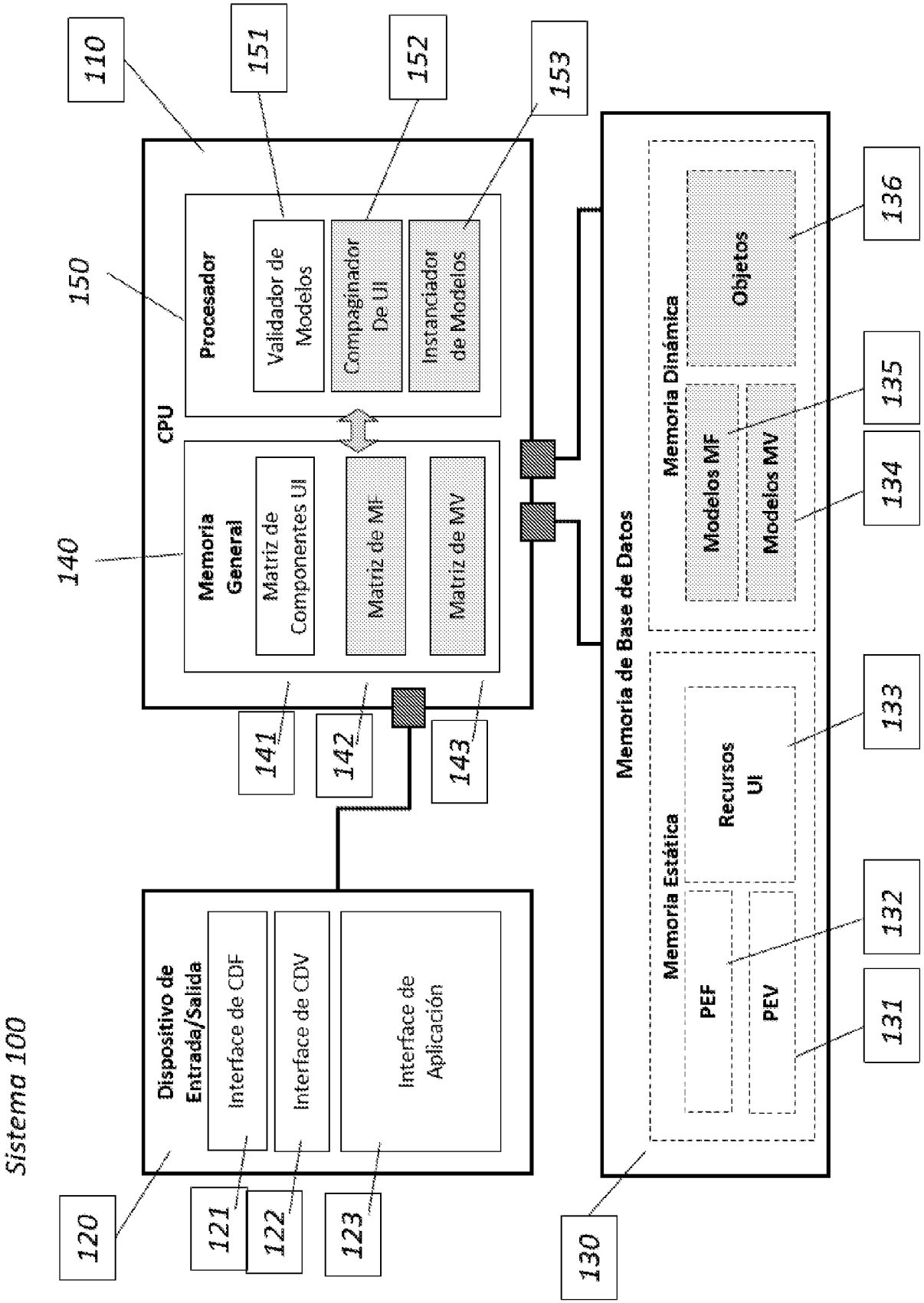

Sistema 100
110
151
152
153
150
Procesador
Validador de Modelos
Compaginador De UI
Instanciador de Modelos
CPU
140
Memoria General
Matriz de Componentes UI
Matriz de MF
Matriz de MV
141
142
143
Memoria de Base de Datos
Memoria Dinámica
Objetos
136
Modelos MF
135
Modelos MV
134
Memoria Estática
Recursos UI
133
PEF
132
PEV
131
130
Dispositivo de Entrada/Salida
Interface de CDF
Interface de CDV
Interface de Aplicación
120
121
122
123

FIG. 1

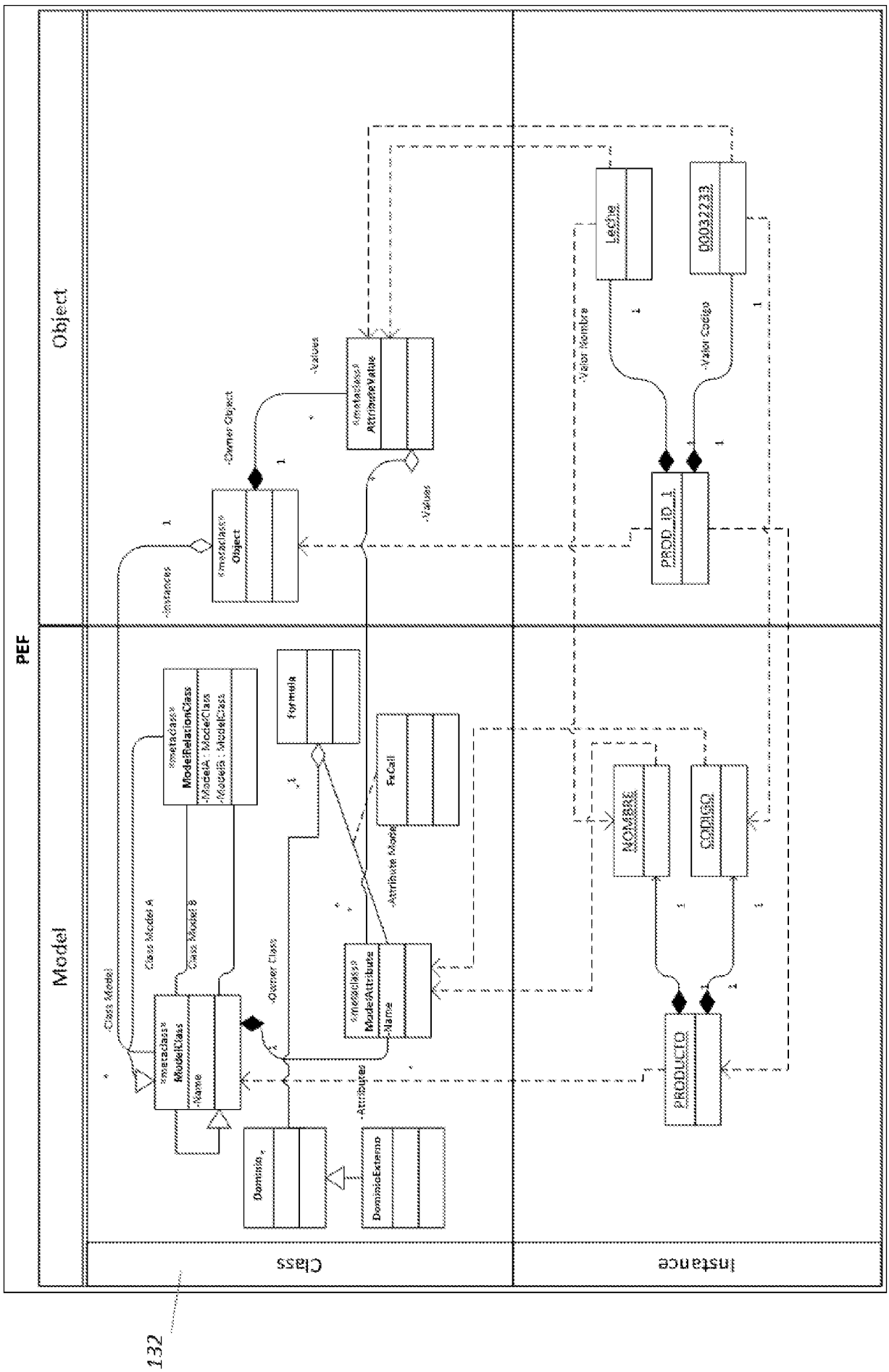

PEF
Object
Model
Class
Instance
«metaclass» Object
«metaclass» AttributeValue
«metaclass» ModelClass
«metaclass» ModelRelationClass
«metaclass» ModelAttribute
Formula
FxCall
Dominio
DominioExterno
PRODUCTO
PROD_ID_1
NOMBRE
CODIGO
Leche
00032233
Owner Object
Values
Instances
Owner Class
Attributes
Class Model
Class Model A
Class Model B
Valor Nombre
Valor Codigo
132

FIG. 1A

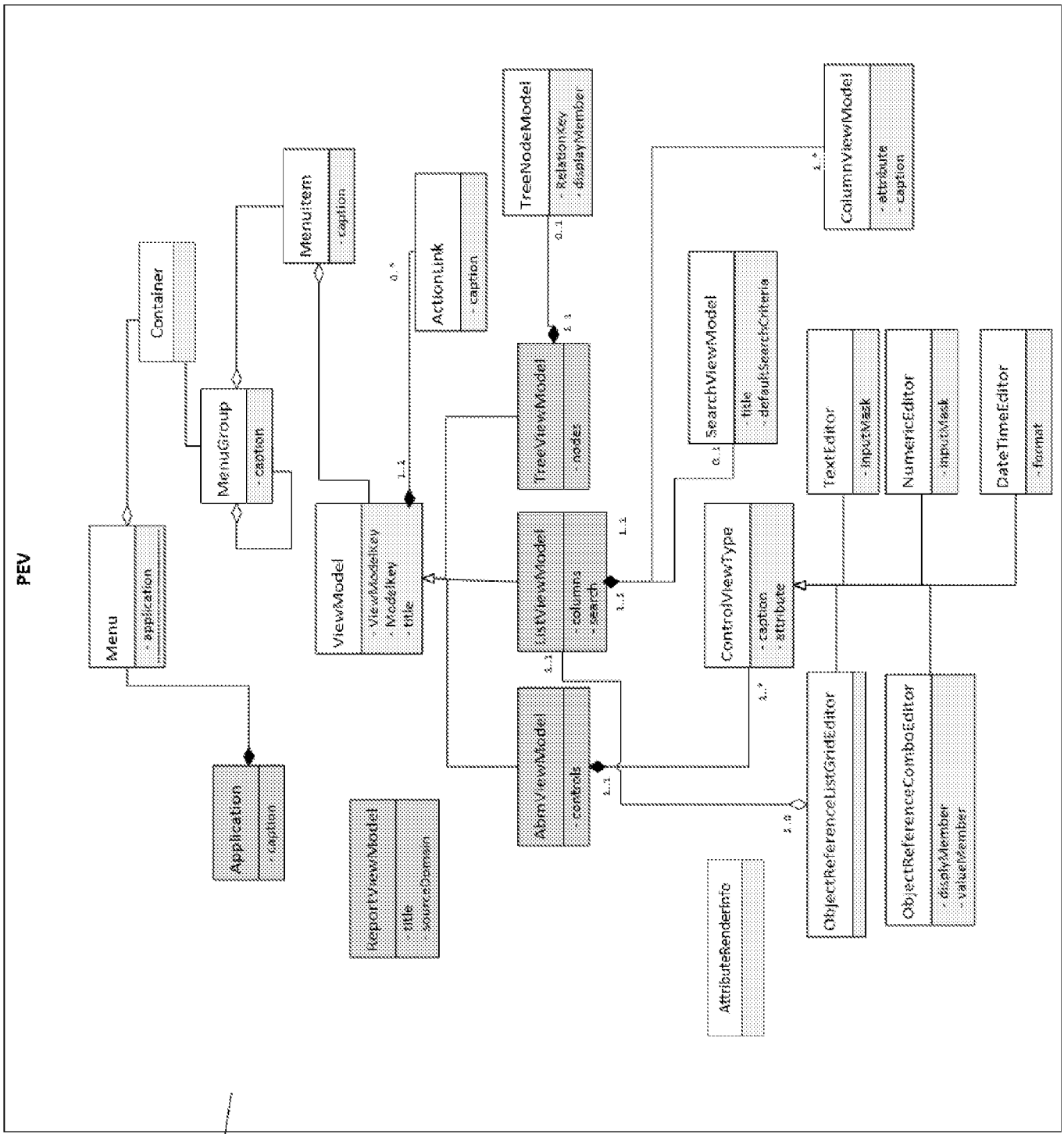

PEV
Container
Menu
- application
MenuGroup
- caption
MenuItem
- caption
Application
- caption
ViewModel
- ViewModelKey
- ModelKey
- title
ActionLink
- caption
TreeNodeModel
- RelationKey
- displayMember
TreeViewModel
- nodes
ListViewModel
- columns
- search
AbmViewModel
- controls
ReportViewModel
- title
- sourceDomain
SearchViewModel
- title
- defaultSearchCriteria
ControlViewType
- caption
- attribute
ColumnViewModel
- attribute
- caption
TextEditor
- inputMask
NumericEditor
- inputMask
DateTimeEditor
- format
ObjectReferenceListGridEditor
ObjectReferenceComboEditor
- displayMember
- valueMember
AttributeRenderInfo
131

FIG. 1B

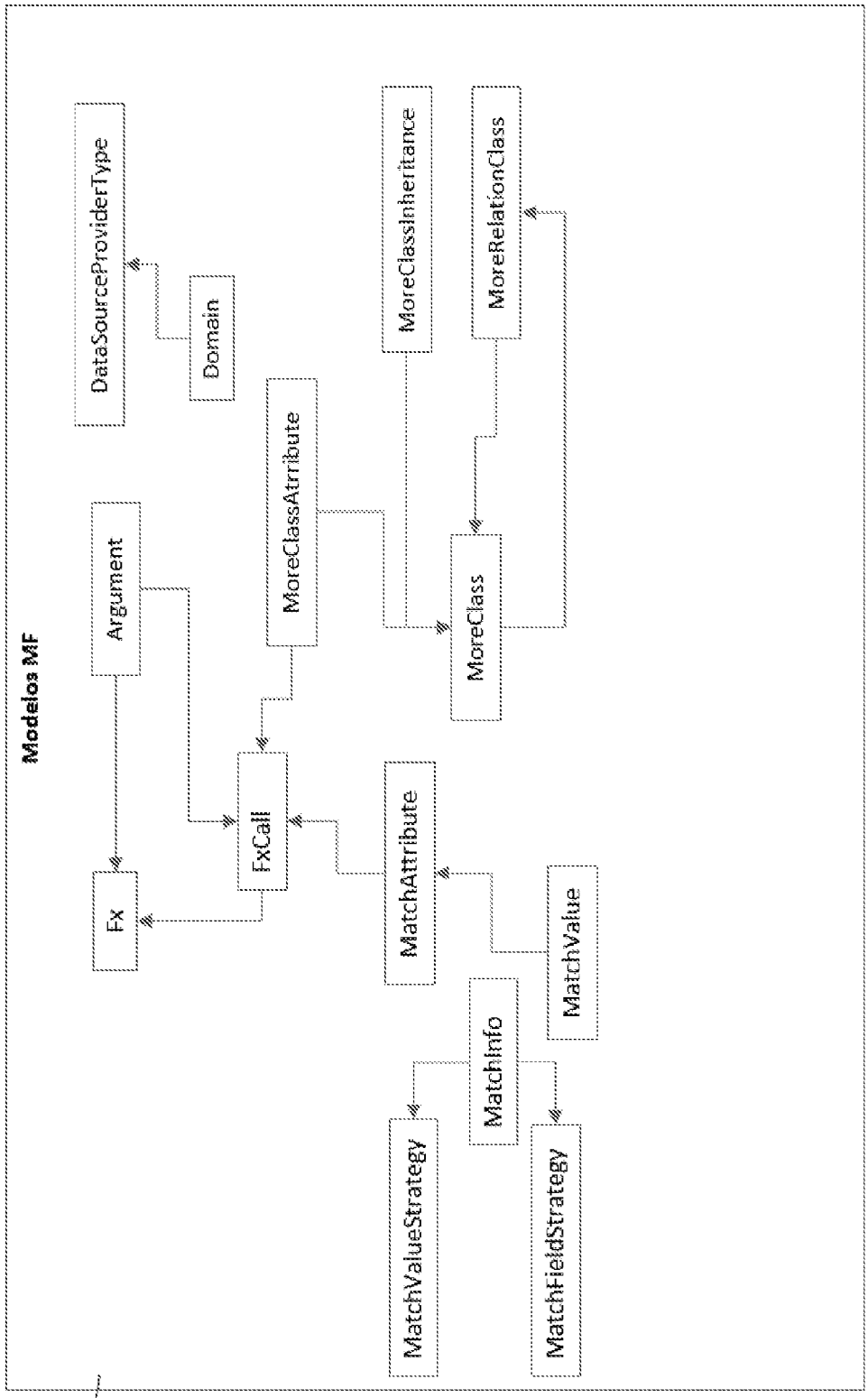

Modelos MF
DataSourceProviderType
Domain
Argument
MoreClassAtrribute
MoreClassInheritance
MoreRelationClass
MoreClass
Fx
FxCall
MatchAttribute
MatchInfo
MatchValue
MatchValueStrategy
MatchFieldStrategy
134

FIG. 1C

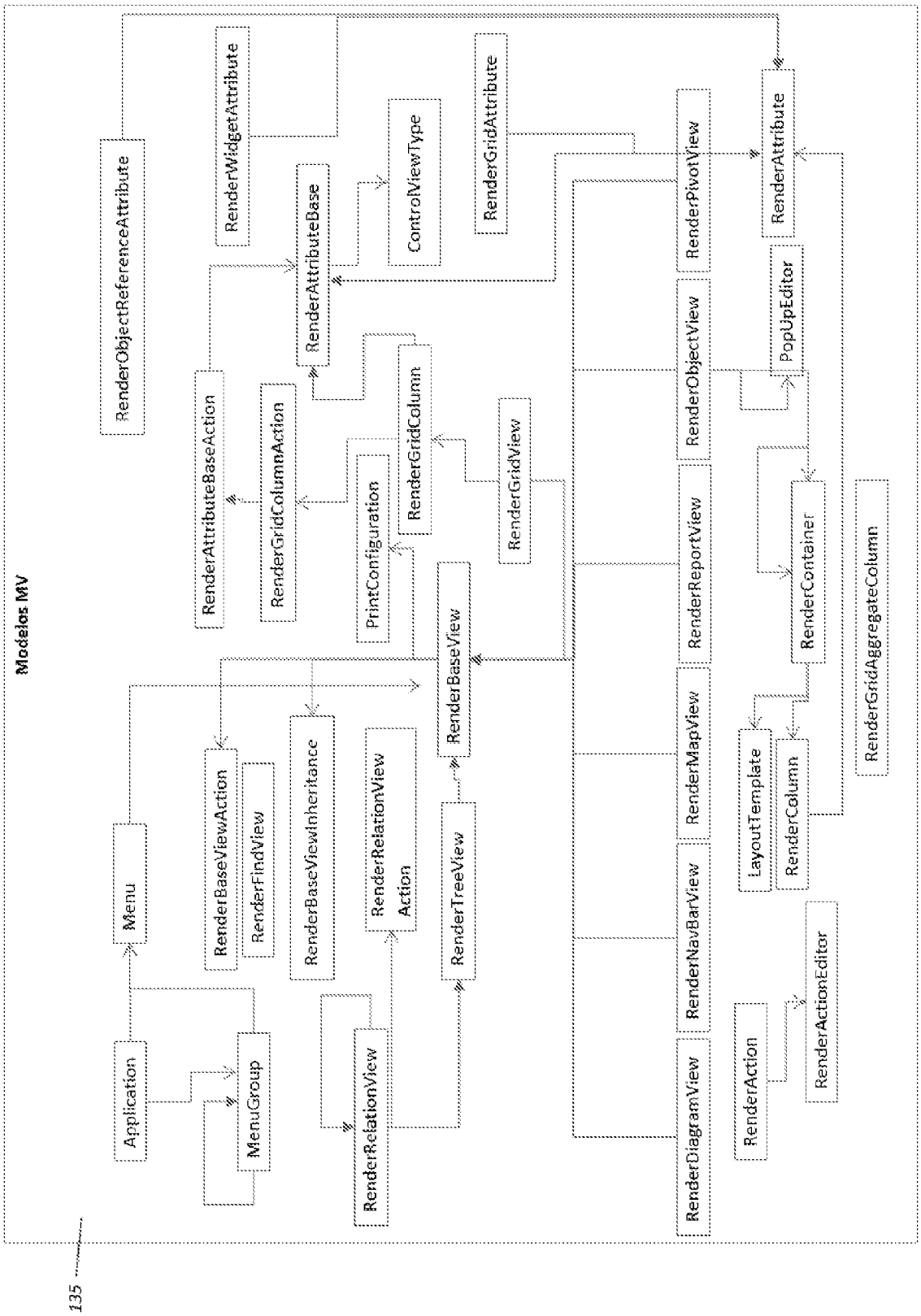

Modelos MV
RenderObjectReferenceAttribute
RenderWidgetAttribute
RenderAttributeBase
ControlViewType
RenderGridAttribute
RenderAttribute
RenderAttributeBaseAction
RenderGridColumnAction
PrintConfiguration
RenderGridColumn
PopUpEditor
RenderPivotView
RenderObjectView
RenderGridView
RenderBaseView
RenderReportView
RenderContainer
Menu
RenderBaseViewAction
RenderFindView
RenderBaseViewInheritance
RenderRelationView Action
RenderTreeView
RenderMapView
LayoutTemplate
RenderGridAggregateColumn
Application
MenuGroup
RenderRelationView
RenderNavBarView
RenderColumn
RenderAction
RenderActionEditor
RenderDiagramView
135

FIG. 1D

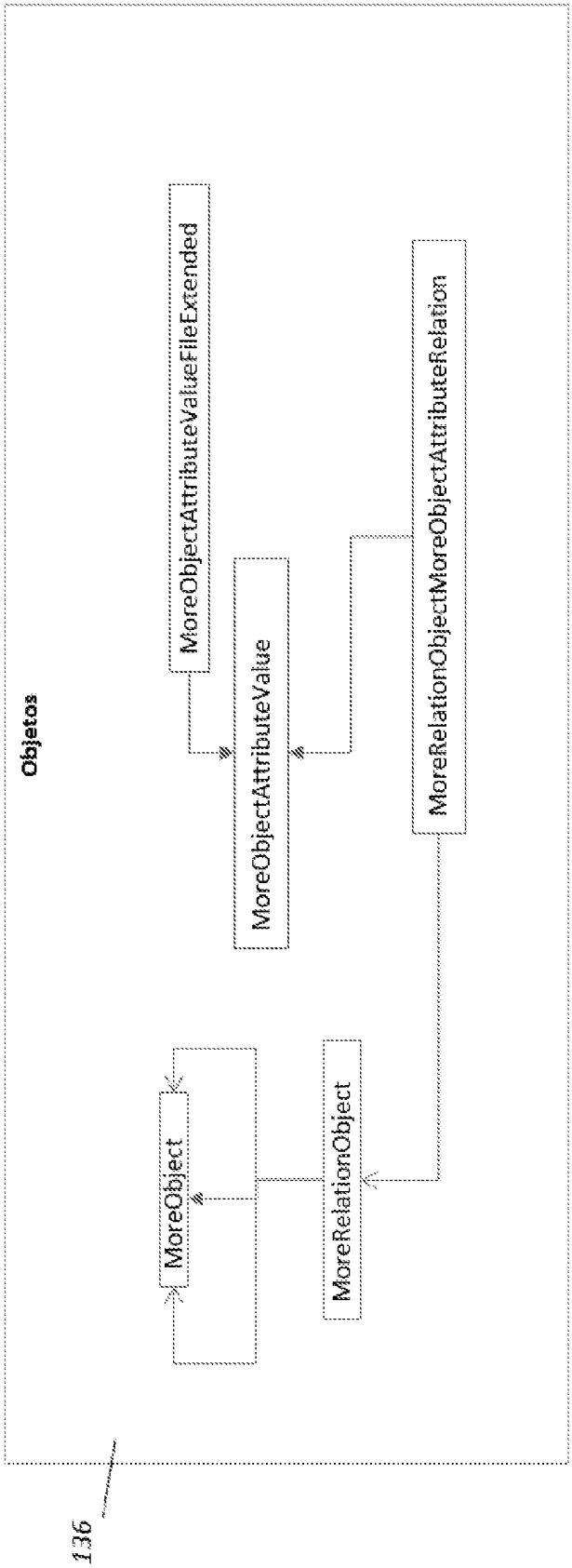

Objetos
MoreObjectAttributeValueFileExtended
MoreObjectAttributeValue
MoreRelationObjectMoreObjectAttributeRelation
MoreObject
MoreRelationObject
136

FIG. 1E

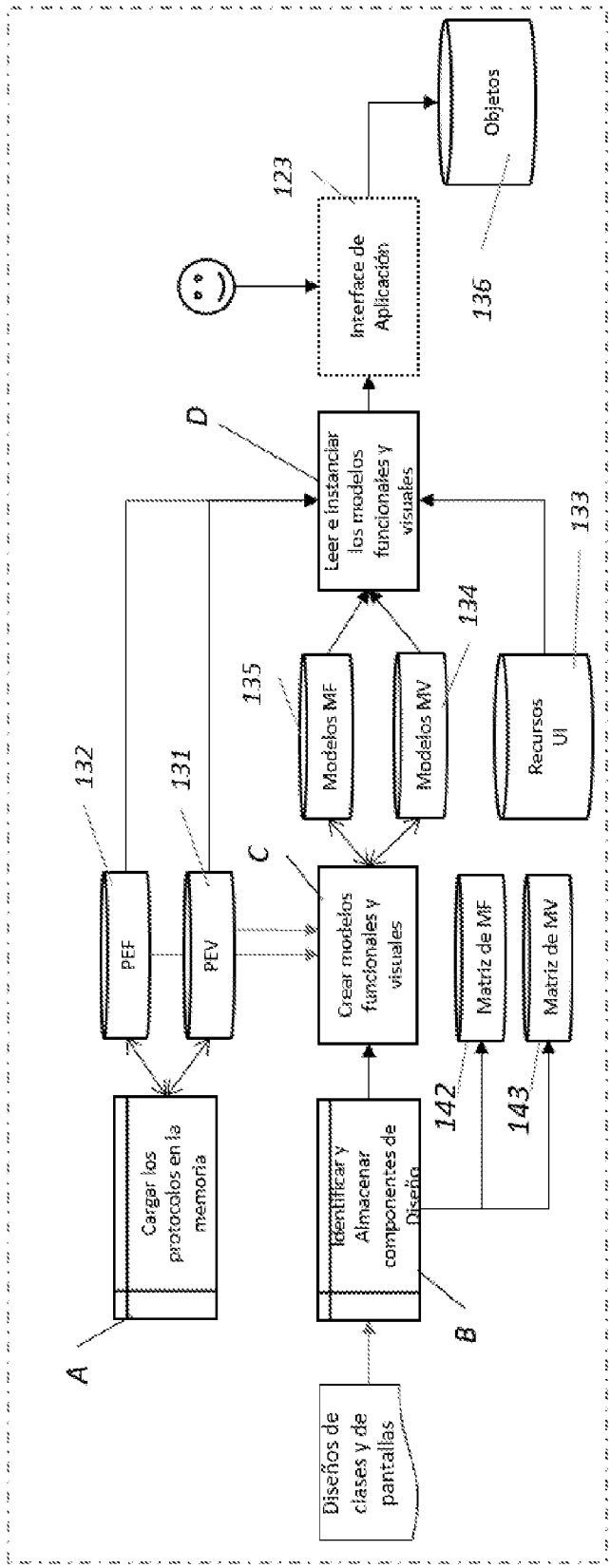

Diseños de clases y de pantallas
Cargar los protocolos en la memoria
Identificar y Almacenar componentes de Diseño
Crear modelos funcionales y visuales
PEF
PEV
Matriz de MF
Matriz de MV
Modelos MF
Modelos MV
Recursos UI
Leer e instanciar los modelos funcionales y visuales
Interface de Aplicación
Objetos
A
B
C
D
131
132
133
134
135
136
142
143

FIG. 2

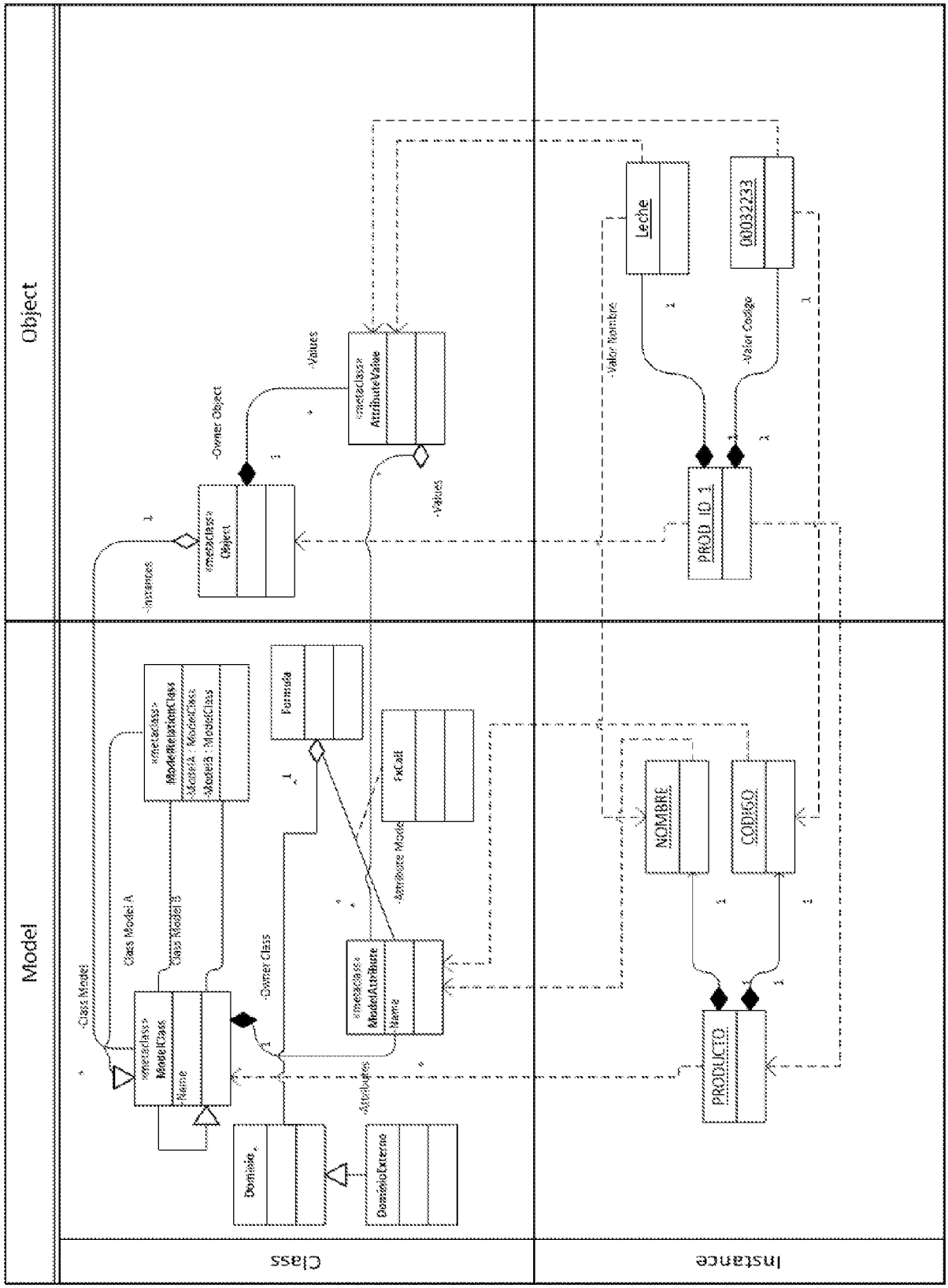

Object
Model
Class
Instance
«metaclass» Object
«metaclass» AttributeValue
«metaclass» ModelClass
«metaclass» ModelRelationClass
«metaclass» ModelAttribute
Formula
FxCall
Dominio
DominioExterno
PROD ID 1
Leche
00032233
PRODUCTO
NOMBRE
CODIGO
-Owner Object
-Values
-instances
-Class Model
-Class Model A
-Class Model B
-Owner Class
-Attributes
-Valor Nombre
-Valor Codigo

FIG. 2A

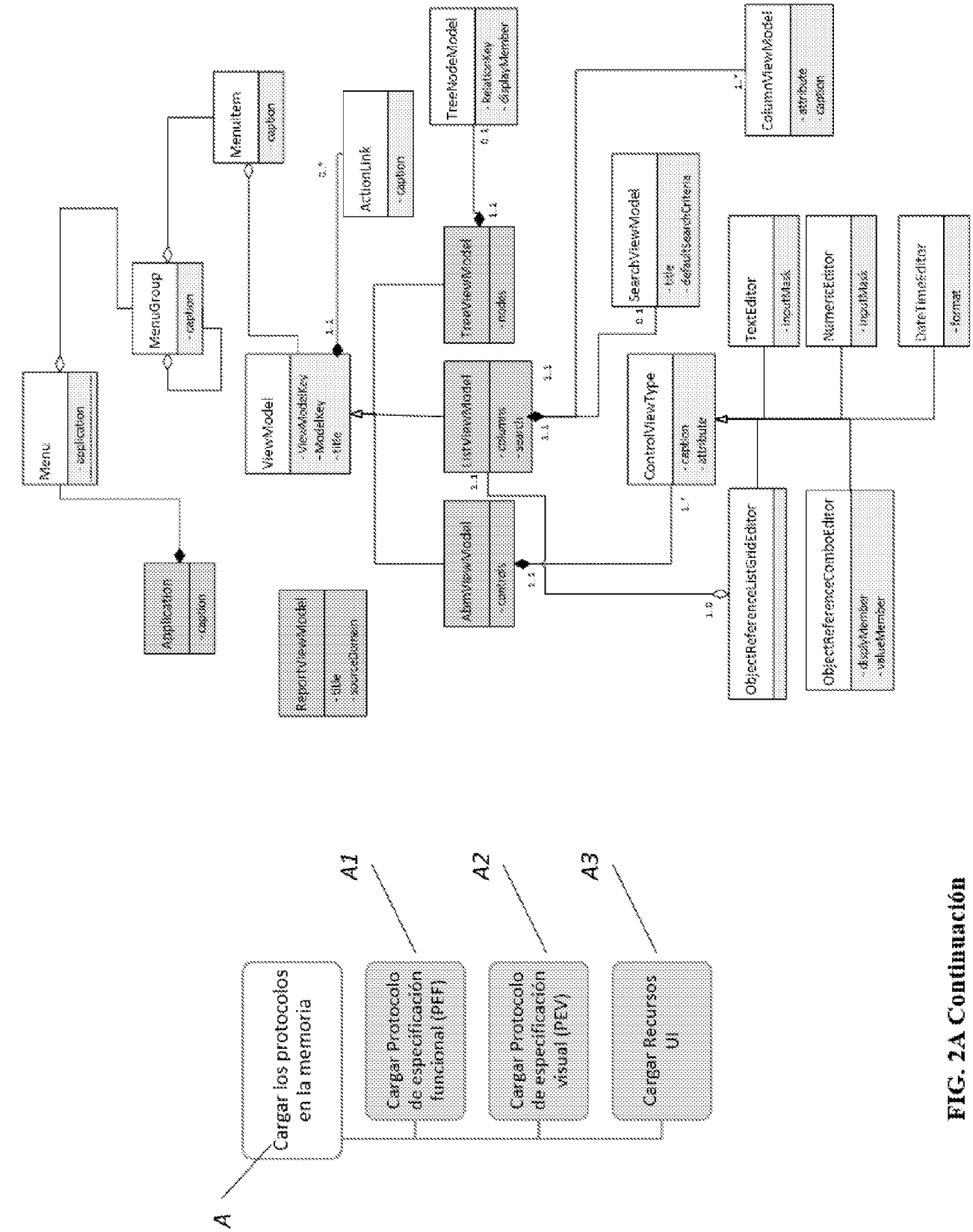


FIG. 2A Continuación

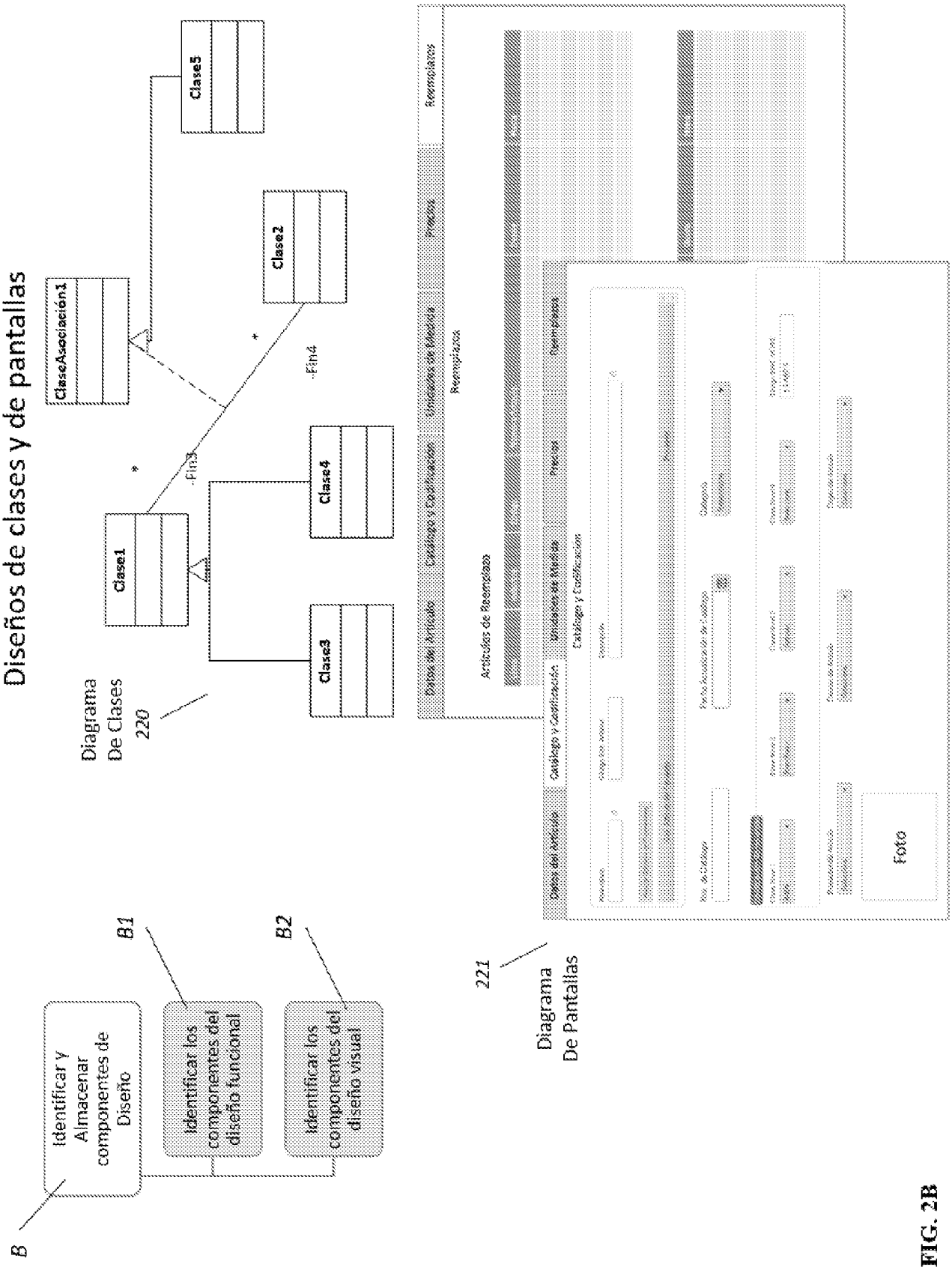
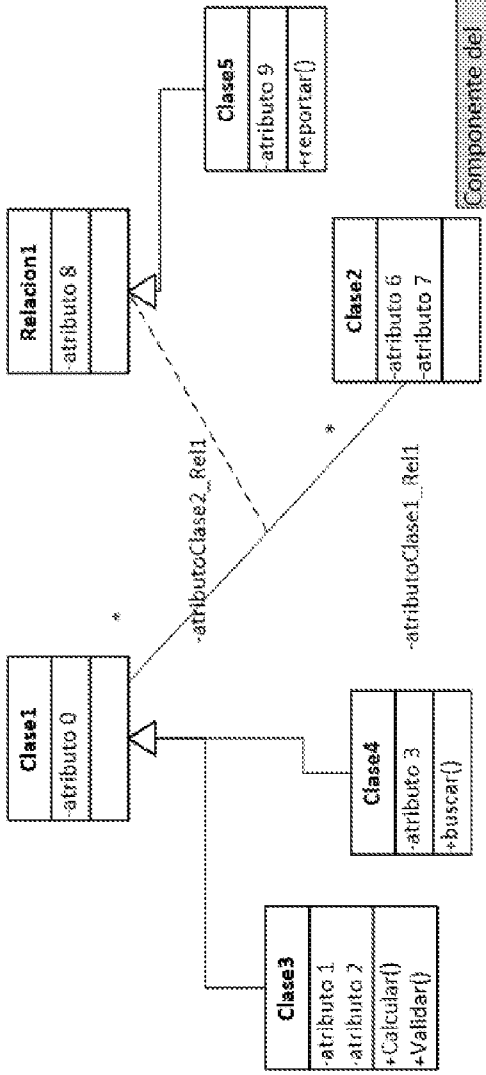
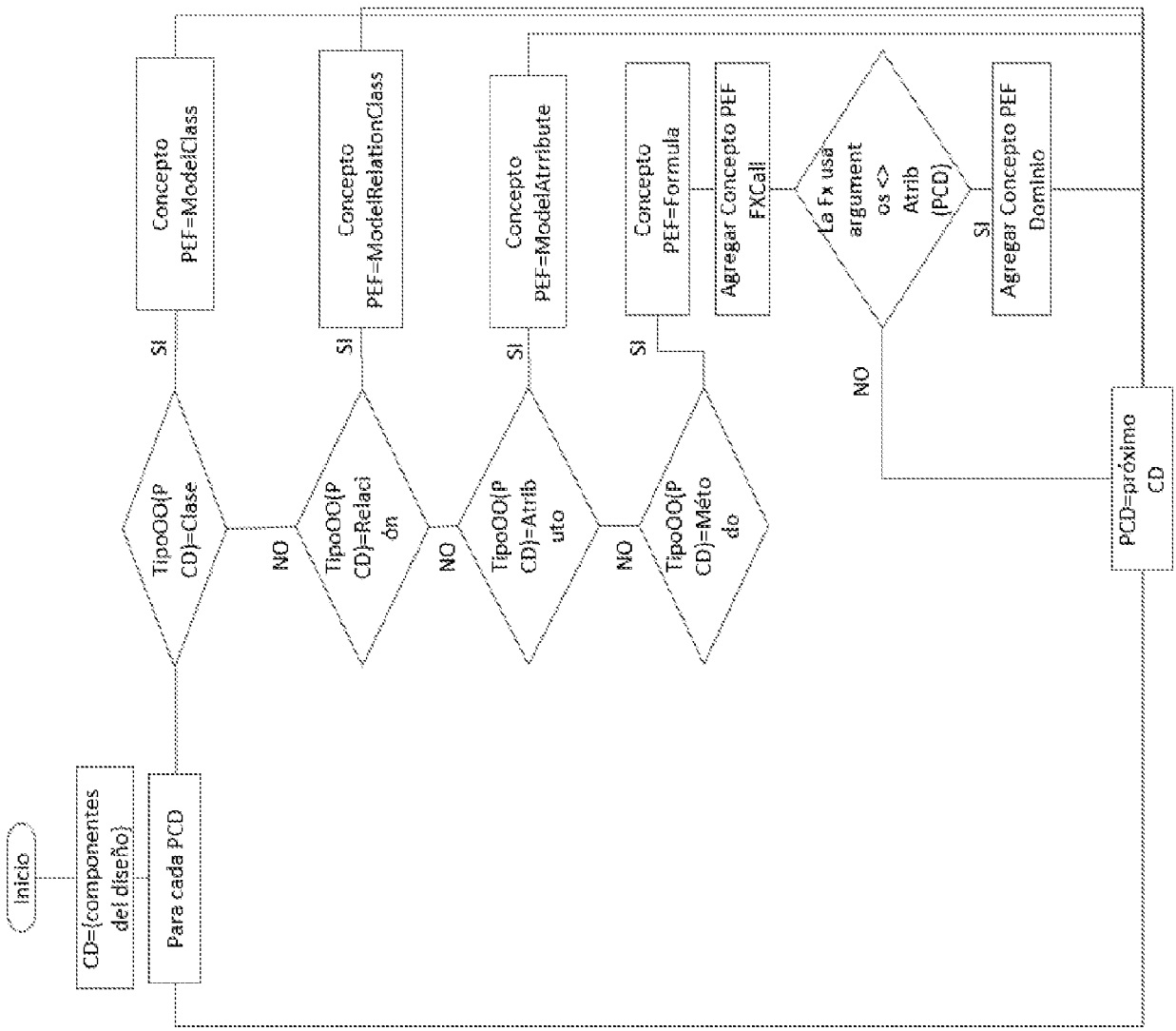

Diseños de clases y de pantallas
Clase5
ClaseAsociación1
Clase2
Clase1
Clase4
Clase3
Diagrama
De Clases
220
Datos del Artículo
Catálogo y Codificación
Unidades de Medida
Precios
Reemplazos
Artículos de Reemplazos
Foto
221
Diagrama
De Pantallas
B
Identificar y
Almacenar
componentes de
Diseño
B1
Identificar los
componentes del
diseño funcional
B2
Identificar los
componentes del
diseño visual

FIG. 2B



Componente del diseño	TipoOO	Nombre
Clase 1	Clase	Clase 1
	Atributo	Atributo0
	Atributo	atributoClase2_Rel1
Clase 2	Clase	Clase 2
	Atributo	Atributo6
	Atributo	Atributo7
	Atributo	atributoClase1_Rel1
Clase 3	Clase	Clase 3
	Atributo	Atributo1
	Atributo	Atributo2
	Método	Calcular
	Método	Validar
Clase 4	Clase	Clase 4
	Atributo	Atributo3
	Método	Buscar
Clase 5	Clase	Clase 5
	Atributo	Atributo9
	Método	Reportar
Relacion1	Atributo	Atributo 8
	Atributo	atributoClase1_Rel1
	Atributo	atributoClase2_Rel1
	Relacion asociación	Relacion1

FIG. 2B1


Inicio
CD={componentes del diseño}
Para cada PCD
TipoOO(PCD)=Clase
SI
Concepto PEF=ModelClass
NO
TipoOO(PCD)=Relación
SI
Concepto PEF=ModelRelationClass
NO
TipoOO(PCD)=Atributo
SI
Concepto PEF=ModelAtrribute
NO
TipoOO(PCD)=Método
SI
Concepto PEF=Formula
Agregar Concepto PEF FXCall
La fx usa argumentos <> Atrib (PCD)
SI
Agregar Concepto PEF Dominio
NO
PCD=próximo CD

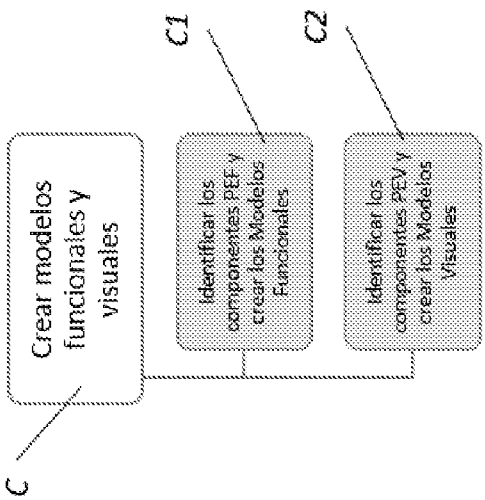

C
Crear modelos funcionales y visuales
C1
Identificar los componentes PEF y crear los Modelos Funcionales
C2
Identificar los componentes PEV y crear los Modelos Visuales

FIG. 2C

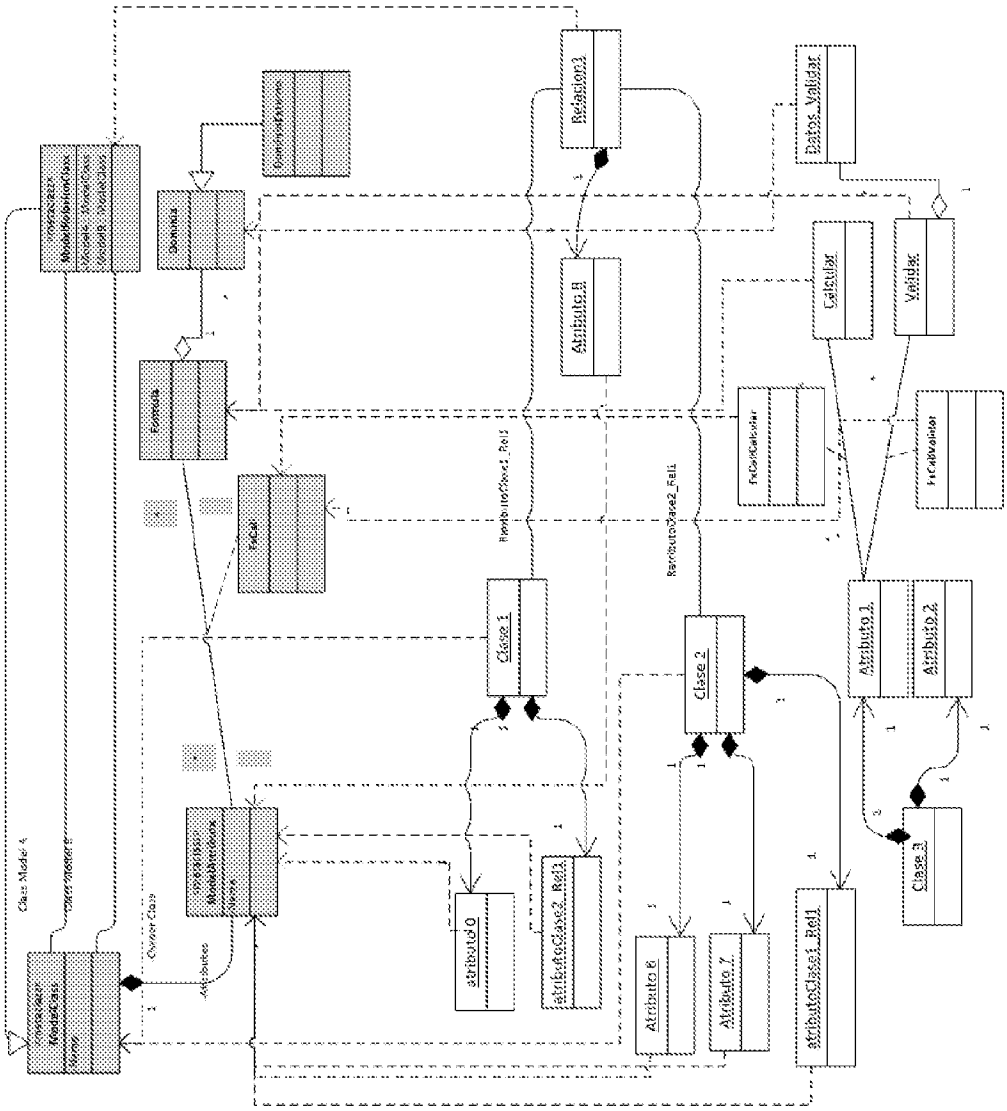

Relacion3
Datos_Validar
Calcular
Validar
Atributo 8
Clase 1
Clase 2
Clase 3
Atributo 1
Atributo 2
atributo 0
atributoClase2_Rel1
Atributo 6
Atributo 7
atributoClase1_Rel1
Class Model 1
Owner Class
Clase5
Relacion3
Clase8
Clase1
Clase6
Clase4

FIG. 2C1

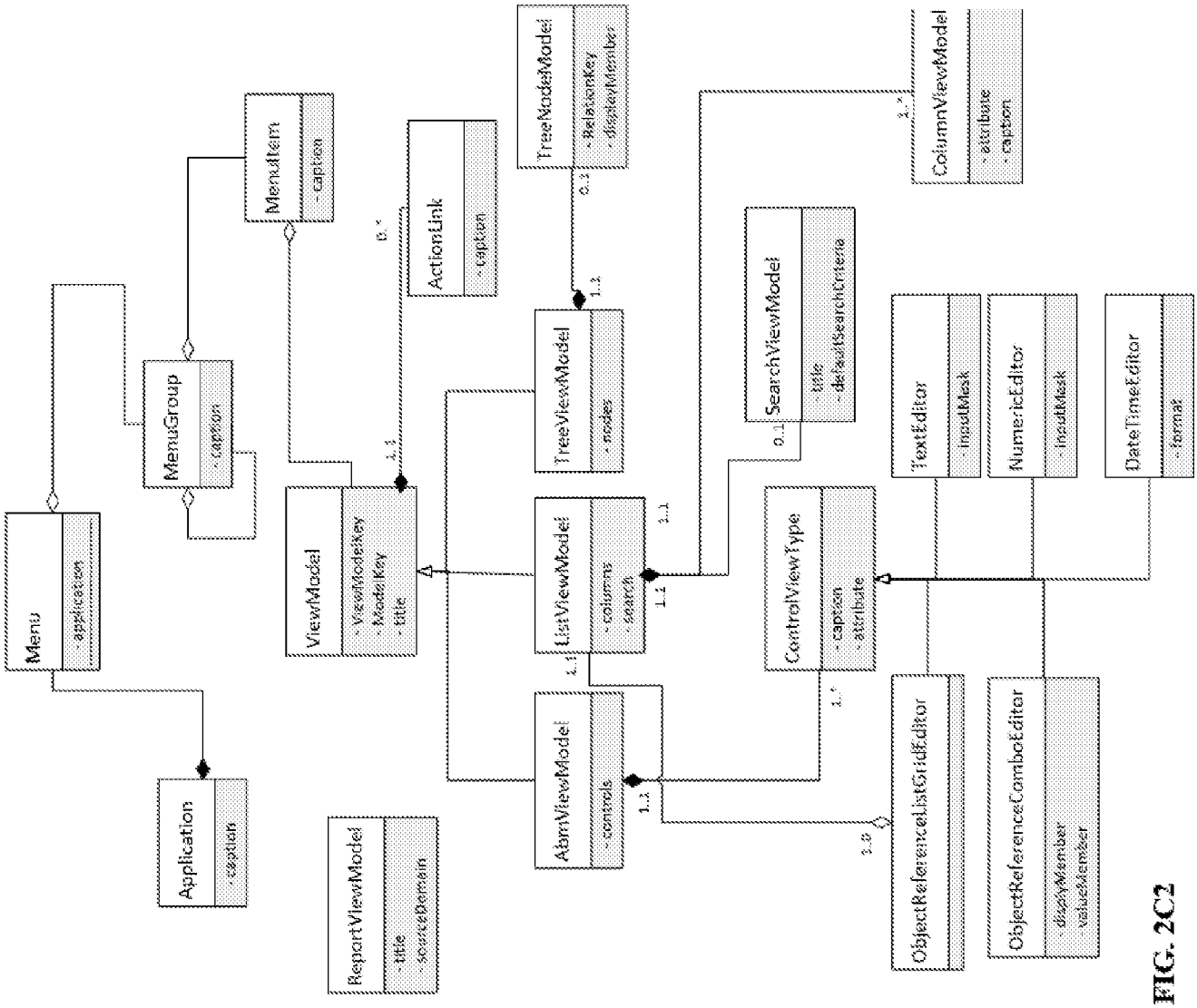
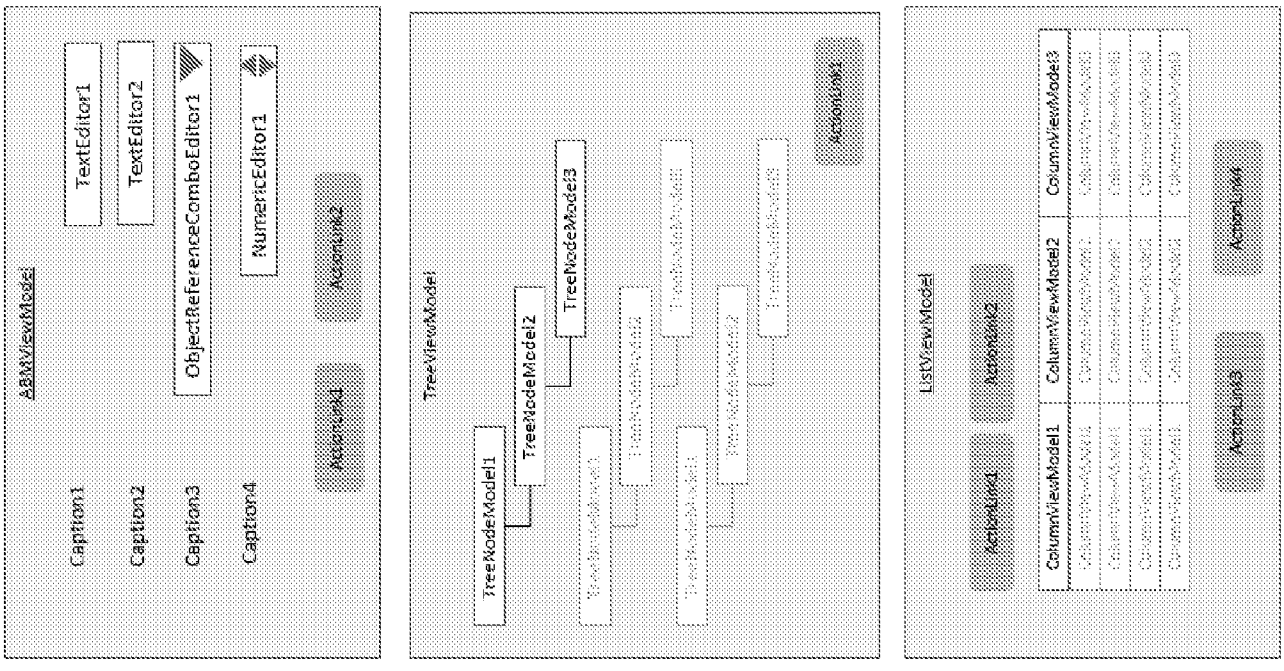

Menu
- application
MenuGroup
- caption
MenuItem
- caption
Application
- caption
ViewModel
- ViewModelKey
- ModelKey
- title
ActionLink
- caption
ReportViewModel
- title
- sourceDomain
TreeViewModel
- nodes
TreeNodeModel
- RelationKey
- displayMember
AbmViewModel
- controls
ListViewModel
- columns
- search
SearchViewModel
- title
- defaultSearchCriteria
ControlViewType
- caption
- attribute
ColumnViewModel
- attribute
- caption
TextEditor
- inputMask
NumericEditor
- inputMask
DateTimeEditor
- format
ObjectReferenceListGridEditor
ObjectReferenceComboEditor
- displayMember
- valueMember

FIG. 2C2


ABMViewModel
Caption1
Caption2
Caption3
Caption4
TextEditor1
TextEditor2
ObjectReferenceComboEditor1
NumericEditor1
Action1
Action2
Action3
Action4
TreeViewModel
TreeNodeModel1
TreeNodeModel2
TreeNodeModel3
Action1
ListViewModel
Action1
Action2
Action3
Action4
ColumnViewModel1
ColumnViewModel2
ColumnViewModel3

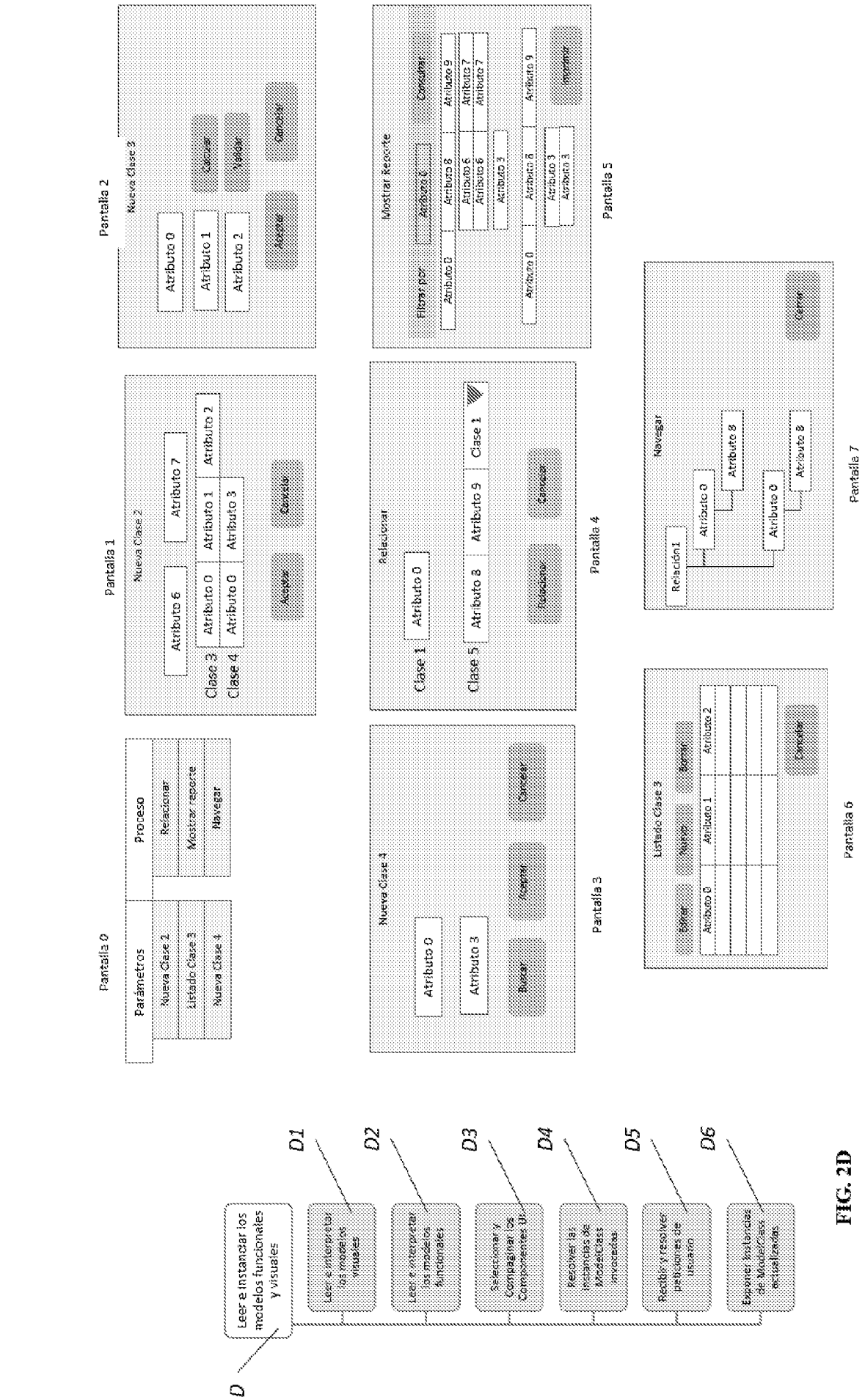

Pantalla 0
Parámetros
Proceso
Nueva Clase 2
Listado Clase 3
Nueva Clase 4
Relacionar
Mostrar reporte
Navegar
Pantalla 1
Nueva Clase 2
Atributo 6
Atributo 7
Clase 3
Clase 4
Atributo 0
Atributo 1
Atributo 2
Atributo 3
Aceptar
Cancelar
Pantalla 2
Nueva Clase 3
Atributo 0
Atributo 1
Atributo 2
Calcular
Validar
Aceptar
Cancelar
Pantalla 3
Nueva Clase 4
Atributo 0
Atributo 3
Borrar
Copiar
Cancelar
Pantalla 4
Relacionar
Clase 1
Clase 5
Atributo 0
Atributo 8
Atributo 9
Relacionar
Cancelar
Pantalla 5
Mostrar Reporte
Filtrar por
Atributo 0
Atributo 6
Atributo 7
Atributo 8
Atributo 9
Atributo 3
Imprimir
Pantalla 6
Listado Clase 3
Borrar
Nuevo
Atributo 0
Atributo 1
Atributo 2
Cancelar
Pantalla 7
Navegar
Relación1
Atributo 0
Atributo 8
Cerrar
D
Leer e instanciar los modelos funcionales y visuales
D1
Leer e interpretar los modelos visuales
D2
Leer e interpretar los modelos funcionales
D3
Seleccionar y compaginar los Componentes UI
D4
Resolver las instancias de ModelClass invocadas
D5
Recibir y resolver peticiones de usuario
D6
Exponer instancias de ModelClass actualizadas

FIG. 2D

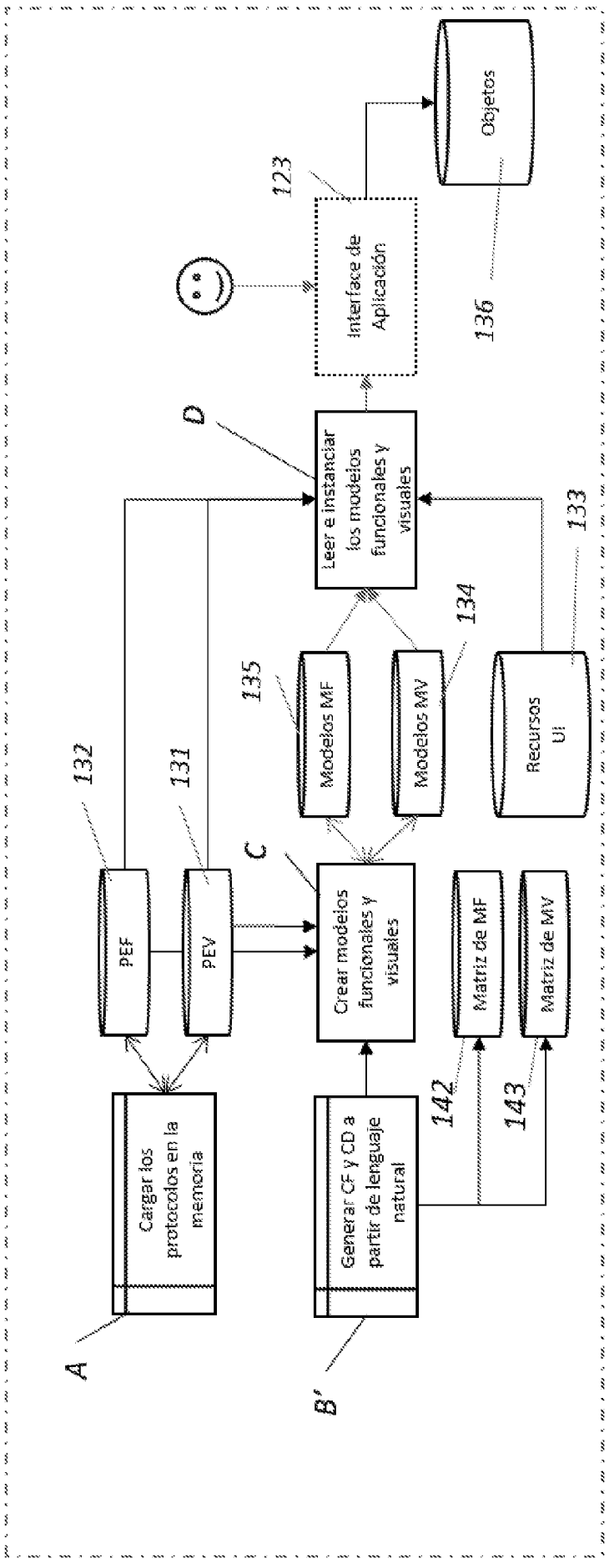

Objetos
123
Interface de Aplicación
136
D
Leer e instanciar los modelos funcionales y visuales
133
134
135
Modelos MF
Modelos MV
Recursos UI
132
131
C
PEF
PEV
Crear modelos funcionales y visuales
Matriz de MF
Matriz de MV
Cargar los protocolos en la memoria
Generar CF y CD a partir de lenguaje natural
142
143
A
B'

FIG. 3A

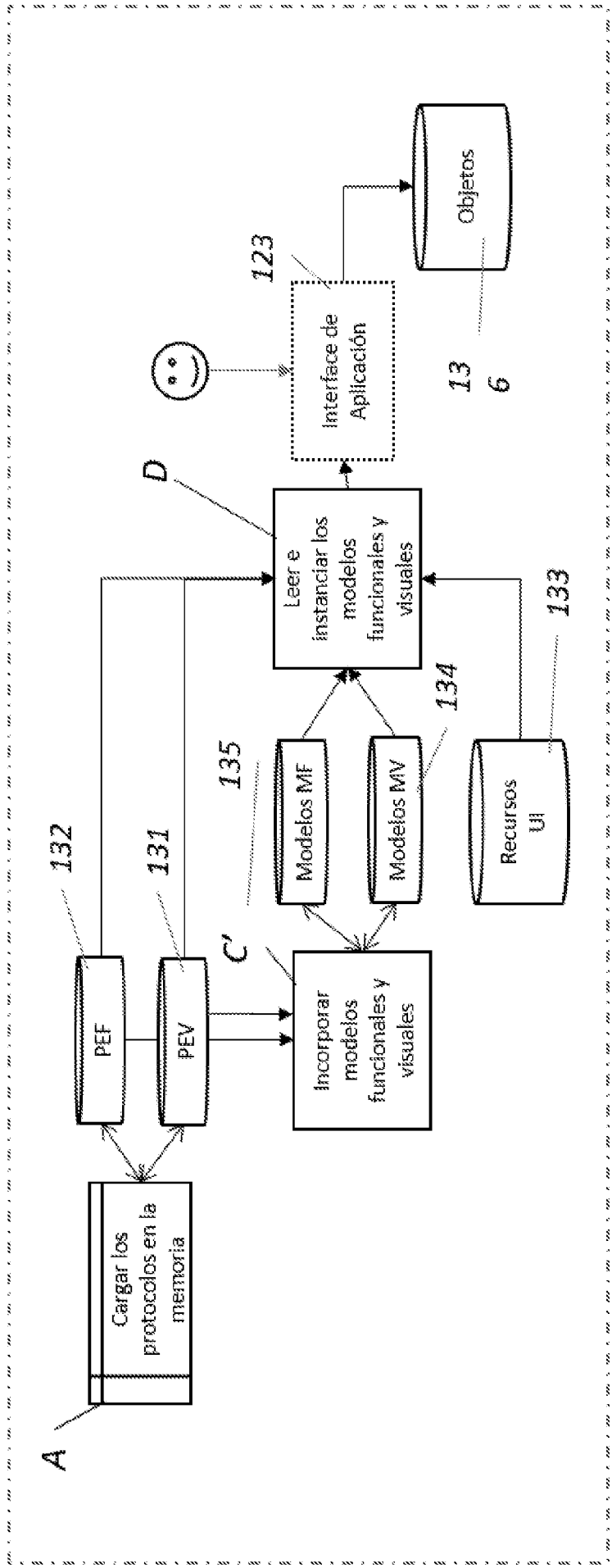

Objetos
123
Interface de Aplicación
13
6
D
Leer e instanciar los modelos funcionales y visuales
133
134
135
Modelos MF
Modelos MV
Recursos UI
132
131
C'
PEF
PEV
Incorporar modelos funcionales y visuales
Cargar los protocolos en la memoria
A

FIG. 3B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB2016/052806

A. CLASSIFICATION OF SUBJECT MATTER

(CIP) G06F 9/45 (2016.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

(CIP) G06F 9/00, G06F 9/45 ; (CPC) G06F 9/44521, G06F 9/44568

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPOQUE, THOMSON, GOOGLE, ESP@CENET, INAPI

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2011/0088011 A1 (VERMEG SARL) 14 April 2011, Paragraph 0010, 0115, 0122, 0135, 0362, 0405, 0413, 0419, 0723, 0724, 1143, 1218. Fig, 24B, claim 1, table 11.	1-11
Y	US 6289513 B1 (BENTWICH, I.) 11 September 2001. Col. 1, lín. 42-49; col. 7, lín. 65-67; col. 8, lín. 46-49; col. 12, lín. 5-7; col. 28, lín. 22-27; col. 59, lín. 37-40, 44-53; col. 60, lín. 20-26. Fig, 12, 20A, claims 1, 38.	1-4, 9-10
Y	US 2009/0183092 A1 (BRITESOFT SOLUTIONS (M) SDN BHD) 16 July 2009. Paragraph 0001, 0006, 0008, 0015, 0017, 0018, 0019. claims 5, 6, 9.	1-11
A	US 8255869 B2 (ROCKWELL AUTOMATION TECHNOLOGIES, INC.) 28 August 2012. abstract. Col. 3, lín. 36-41; Col. 4, lín. 19-22; Col. 5, lín. 44-52; Col. 6, lín. 33-39; Col. 7, lín. 13-22, 29-39; Col. 10, lín. 30-42; Col. 12, lín. 16-27. claims 1, 2.	

☒ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 August 2016 (22.08.16)	13 Septembre 2016 (13.09.2016)
Name and mailing address of the ISA/CL Facsimile No.	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB2016/052806

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2012/0110558 A1 (MICROSOFT CORPORATION) 03 Mayo 2012. Párr. 0018, 0027, 0029. Reiv. 1, 19. Fig 3, 4.	

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB2016/052806

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☒ Claims Nos.: 1.a, 6.b
because they relate to subject matter not required to be searched by this Authority, namely:

These claims relate to a subject matter that is considered by this International Searching Authority to be affected by PCT Rules 39.1(iii) and 67.1(iii), concerning schemes, rules or methods of doing business, performing purely mental acts or playing games. In conclusion, no expert opinion will be established in respect of novelty, inventive step and industrial applicability for these claims (PCT Article 17(2)(a)(b)).

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.

US 2011/0088011 (A1)	14-04-2011	AU 2010/308132 (A1)	10-05-2012
		CA 2777443 (A1)	21-04-2011
		CN 102656557 (A)	05-09-2012
		EP 2488941 (A1)	22-08-2012
		MA 33799 (B1)	03-12-2012
		SG 10201500690R (A)	29-04-2015
		US 2014/109037 (A1)	17-04-2014
		WO 2011/045634 (A1)	21-04-2011
		ZA 2012/02757 (B)	26-06-2013
US 6289513 (B1)	11-09-2001	AU 4946600 (A)	18-12-2000
		EP 1247177 (A2)	09-10-2002
		JP 2003/522991 (A)	29-07-2003
		WO 0074394 (A2)	07-12-2000
US 2009/0183092 (A1)	16-07-2009	AU 2008/229743 (A1)	23-04-2009
		GB 2453437 (A)	08-04-2009
		WO 2009/045094 (A2)	13-08-2009
US 8255869 (B2)	28-08-2012	US 2009/327991 (A1)	31-12-2009
US 2012/0110558 (A1)	03-05-2012	NONE	

INFORME DE BÚSQUEDA INTERNACIONAL

Solicitud internacional N°
PCT/IB2016/052806

A. CLASIFICACIÓN DEL OBJETO DE LA SOLICITUD

(CIP) G06F 9/45 (2016.01)

De acuerdo con la Clasificación Internacional de Patentes (CIP) o según la clasificación nacional y CIP.

B. SECTORES COMPRENDIDOS POR LA BÚSQUEDA

Documentación mínima buscada (sistema de clasificación seguido de los símbolos de clasificación)

(CIP) G06F 9/00, G06F 9/45 ; (CPC) G06F 9/44521, G06F 9/44568

Otra documentación consultada, además de la documentación mínima, en la medida en que tales documentos formen parte de los sectores comprendidos por la búsqueda

Bases de datos electrónicas consultadas durante la búsqueda internacional (nombre de la base de datos y, si es posible, términos de búsqueda utilizados) EPOQUE, THOMSON, GOOGLE, ESP@CENET, INAPI

C. DOCUMENTOS CONSIDERADOS RELEVANTES

Categoría*	Documentos citados, con indicación, si procede, de las partes relevantes	Relevante para las reivindicaciones N°
Y	US 2011/0088011 A1 (VERMEG SARL) 14 Abril 2011. Párr. 0010, 0115, 0122, 0135, 0362, 0405, 0413, 0419, 0723, 0724, 1143, 1218. Fig 24B. Reiv. 1. Tabla 11.	1-11
Y	US 6289513 B1 (BENTWICH, I.) 11 Septiembre 2001. Col. 1, lín. 42-49; col. 7, lín. 65-67; col. 8, lín. 46-49; col. 12, lín. 5-7; col. 28, lín. 22-27; col. 59, lín. 37-40, 44-53; col. 60, lín. 20-26. Fig 12, 20A. Reiv. 1, 38.	1-4, 9-10
Y	US 2009/0183092 A1 (BRITESOFT SOLUTIONS (M) SDN BHD) 16 Julio 2009. Párr. 0001, 0006, 0008, 0015, 0017, 0018, 0019. Reiv. 5, 6, 9.	1-11
A	US 8255869 B2 (ROCKWELL AUTOMATION TECHNOLOGIES, INC.) 28 Agosto 2012. Resumen. Col. 3, lín. 36-41; Col. 4, lín. 19-22; Col. 5, lín. 44-52; Col. 6, lín. 33-39; Col. 7, lín. 13-22, 29-39; Col. 10, lín. 30-42; Col. 12, lín. 16-27. Reiv. 1, 2.	

☒ En la continuación del Recuadro C se relacionan otros documentos ☒ Los documentos de familias de patentes se indican en el Anexo

* Categorías especiales de documentos citados:

"A" documento que define el estado general de la técnica no considerado como particularmente relevante.

"E" solicitud de patente o patente anterior pero publicada en la fecha de presentación internacional o en fecha posterior.

"L" documento que puede plantear dudas sobre una reivindicación de prioridad o que se cita para determinar la fecha de publicación de otra cita o por una razón especial (como la indicada).

"O" documento que se refiere a una divulgación oral, a una utilización, a una exposición o a cualquier otro medio.

"P" documento publicado antes de la fecha de presentación internacional pero con posterioridad a la fecha de prioridad reivindicada.

"T" documento ulterior publicado con posterioridad a la fecha de presentación internacional o de prioridad que no pertenece al estado de la técnica pertinente pero que se cita por permitir la comprensión del principio o teoría que constituye la base de la invención.

"X" documento particularmente relevante; la invención reivindicada no puede considerarse nueva o que implique una actividad inventiva por referencia al documento aisladamente considerado.

"Y" documento particularmente relevante; la invención reivindicada no puede considerarse que implique una actividad inventiva cuando el documento se asocia a otro u otros documentos de la misma naturaleza, cuya combinación resulta evidente para un experto en la materia.

"&" documento que forma parte de la misma familia de patentes.

Fecha en que se ha concluido efectivamente la búsqueda internacional.	Fecha de expedición del informe de búsqueda internacional
22/08/2016 22/agosto/2016	13/09/2016 13/septiembre/2016
Nombre y dirección postal de la Administración encargada de la búsqueda internacional INAPI, Av. Libertador Bernardo O'Higgins 194, Piso 17, Santiago, Chile	Funcionario autorizado ARAYA LARA, Hugo Boris
N° de fax	N° de teléfono 56-2-28870550 56-2-28870551

INFORME DE BÚSQUEDA INTERNACIONAL

Solicitud internacional N°

PCT/IB2016/052806

C (continuación). DOCUMENTOS CONSIDERADOS RELEVANTES

Categoría*	Documentos citados, con indicación, si procede, de las partes relevantes	Relevante para las reivindicaciones N°
A	US 2012/0110558 A1 (MICROSOFT CORPORATION) 03 Mayo 2012. Párr. 0018, 0027, 0029. Reiv. 1, 19. Fig 3, 4.	

INFORME DE BÚSQUEDA INTERNACIONAL

Solicitud internacional N°

PCT/IB2016/052806

Recuadro II **Observaciones cuando se estime que algunas reivindicaciones no pueden ser objeto de búsqueda (continuación del punto 2 de la primera hoja)**

Este informe de búsqueda internacional no se ha realizado en relación a ciertas reivindicaciones según el Artículo 17.2)a) por los siguientes motivos:

1. ☒ Las reivindicaciones N^{os}: 1.a, 6.b
se refieren a un objeto con respecto al cual esta Administración no está obligada a proceder a la búsqueda, a saber:
Estas cláusulas se refieren a una materia que esta Administración considera afectada por las Reglas PCT 39.1(iii) y 67.1(iii), concernientes a planes, principios o métodos para hacer negocios, para actos puramente intelectuales o en materia de juego. En conclusión, ninguna opinión experta será establecida para novedad, nivel inventivo y aplicación industrial de dichas cláusulas (Art.17(2)(a)(b)).

2. ☐ Las reivindicaciones N^{os}:
se refieren a elementos de la solicitud internacional que no cumplen con los requisitos establecidos, de tal modo que no pueda efectuarse una búsqueda provechosa, concretamente:

3. ☐ Las reivindicaciones N^{os}:
son reivindicaciones dependientes y no están redactadas de conformidad con los párrafos segundo y tercero de la Regla 6.4.a).

Recuadro III **Observaciones cuando falta unidad de invención (continuación del punto 3 de la primera hoja)**

La Administración encargada de la búsqueda internacional ha detectado varias invenciones en la presente solicitud internacional, a saber:

1. ☐ Dado que todas las tasas adicionales requeridas han sido satisfechas por el solicitante dentro del plazo, el presente informe de búsqueda de tipo internacional comprende todas las reivindicaciones que pueden ser objeto de búsqueda.

2. ☐ Dado que todas las reivindicaciones que pueden ser objeto de búsqueda podrían serlo sin realizar un esfuerzo que justifique tasas adicionales, esta Administración no requirió el pago de tasas adicionales.

3. ☐ Dado que tan sólo una parte de las tasas adicionales requeridas ha sido satisfecha dentro del plazo por el solicitante, el presente informe de búsqueda de tipo internacional comprende solamente aquellas reivindicaciones respecto de las cuales han sido satisfechas las tasas, concretamente las reivindicaciones N^{os}:

4. ☐ Ninguna de las tasas adicionales requeridas ha sido satisfecha por el solicitante dentro de plazo. En consecuencia, el presente informe de búsqueda de tipo internacional se limita a la invención mencionada en primer término en las reivindicaciones, cubierta por las reivindicaciones N^{os}:

Indicación en cuanto a la protesta

☐ Se acompañó a las tasas adicionales la protesta del solicitante y, en su caso, el pago de una tasa de protesta.

☐ Se acompañó a las tasas adicionales la protesta del solicitante, pero la tasa de protesta aplicable no se pagó en el plazo establecido en el requerimiento.

☐ El pago de las tasas adicionales no ha sido acompañado de ninguna protesta.

INFORME DE BÚSQUEDA INTERNACIONAL

Información relativa a miembros de familias de patentes

Solicitud internacional Nº

PCT/IB2016/052806

Documento de patente citado en Informe de Búsqueda	Fecha de Publicación	Miembro(s) de Familia	Fecha de Publicación
US 2011/0088011 (A1)	14-04-2011	AU 2010/308132 (A1)	10-05-2012
		CA 2777443 (A1)	21-04-2011
		CN 102656557 (A)	05-09-2012
		EP 2488941 (A1)	22-08-2012
		MA 33799 (B1)	03-12-2012
		SG 10201500690R (A)	29-04-2015
		US 2014/109037 (A1)	17-04-2014
		WO 2011/045634 (A1)	21-04-2011
		ZA 2012/02757 (B)	26-06-2013
US 6289513 (B1)	11-09-2001	AU 4946600 (A)	18-12-2000
		EP 1247177 (A2)	09-10-2002
		JP 2003/522991 (A)	29-07-2003
		WO 0074394 (A2)	07-12-2000
US 2009/0183092 (A1)	16-07-2009	AU 2008/229743 (A1)	23-04-2009
		GB 2453437 (A)	08-04-2009
		WO 2009/045094 (A2)	13-08-2009
US 8255869 (B2)	28-08-2012	US 2009/327991 (A1)	31-12-2009
US 2012/0110558 (A1)	03-05-2012	Ninguno	