(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 284 999**

(51) Int. Cl.:
G06F 9/44 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Número de solicitud europea: **03005016 .5**
(86) Fecha de presentación : **05.03.2003**
(87) Número de publicación de la solicitud: **1347376**
(87) Fecha de publicación de la solicitud: **24.09.2003**

(54) Título: **Software para la visualización de objetos estructurables jerárquicamente.**

(30) Prioridad: **18.03.2002 DE 102 11 953**

(45) Fecha de publicación de la mención BOPI:
16.11.2007

(45) Fecha de la publicación del folleto de la patente:
16.11.2007

(73) Titular/es: **SIEMENS AKTIENGESELLSCHAFT**
Wittelsbacherplatz 2
80333 München, DE

(72) Inventor/es: **Herkert, Gebhard**

(74) Agente: **Carvajal y Urquijo, Isabel**

DESCRIPCIÓN

Software para la visualización de objetos estructurables jerárquicamente.

La presente invención hace referencia a un componente de software para la proyección de sistemas de automatización.

Un componente de este tipo se emplea particularmente en el ámbito de la tecnología de la automatización. Para la producción, procesamiento, análisis y aviso de uno de estos programas de automatización se emplean herramientas de programación, ejecutables en un dispositivo de procesamiento de datos.

Para la proyección de instalaciones de automatización existen hoy por hoy diferentes sistemas especiales con almacenamiento independiente de datos. Son ejemplos de estos sistemas: listas de señales para controladores programables (listas de señales SPS) programas secuenciales SPS, programas NC, controles de recepción, sistemas de servicio y supervisión, etc... Para estos sistemas especiales se emplean diferentes ficheros y estructuras de datos. Del mismo modo, frecuentemente los mecanismos de procesamiento se ejecutan y actualizan en múltiples ocasiones, lo que conlleva unos altos costes de desarrollo.

Gracias a la US-A-5 619 638 se conoce un sistema computerizado orientado a objeto con objetos de representación. Los objetos de representación posibilitan una alternativa o también una representación múltiple de los objetos de datos individuales, por ejemplo, se pueden representar alternativamente diagramas de barras y de tartas a partir de un inventario individual de datos.

La US-A-5 644 771 se relaciona con un procedimiento para el manejo de diversos objetos. El modelo de objeto propuesto consiste en tres estructuras de datos clave: la estructura del objeto, la tabla de interfaz y la tabla de métodos.

Gracias a la US-A-6 028 998 se conoce un programa master de aplicación para la construcción de sistemas de automatización para inmuebles. El programa master de aplicación para la automatización de inmuebles define una jerarquía de clases orientada al objeto, en la que un objeto estándar "superclase" define un gran número de objetos estándar diferentes, que pueden componerse, mediante una herramienta apropiada, en sistemas de automatización de inmuebles mayores y más complejos. El objeto estándar contiene un componente visual, que contiene detalles de implementación respecto al "indicador", así como componentes de comando, que contienen procedimientos, para ocasionar determinados controles en otros objetos.

La presente invención se basa en el objetivo de especificar un sistema y/o un procedimiento, que posibilite una proyección uniforme con, al mismo tiempo, una alta consistencia de datos.

Este objetivo se resuelve mediante la combinación de las características indicadas en la Reivindicación 1.

La presente invención se basa en el conocimiento de que, en un sistema de proyección de componentes de software de un sistema de automatización, existen generalmente datos duplicados, teniendo estos datos que introducirse generalmente en múltiples ocasiones, por lo que se produce, además de un coste de tratamiento considerablemente alto, una mayor susceptibilidad al fallo, así como problemas respecto a la consistencia de datos. La arquitectura del software del nuevo sistema se basa, por el contrario, en un componente de software, que presenta una estructura de datos de un árbol de objetos con objetos estructurables jerárquicamente. Una estructura de datos de este tipo posibilita un almacenamiento único de datos para todos los sistemas parciales, como por ejemplo, programas secuenciales, controladores de enlace, visualización en 2D, visualización en 3D, etc. Todos los componentes de software presentan una estructura de datos y base de datos uniformes para todos los componentes de la instalación, ya que todos los componentes están sujetos a un, y un mismo, mecanismo estructural. En conjunto, se origina una estructura única de objetos dentro de un proyecto, de forma que se posibiliten modificaciones también mediante un único mecanismo de copia, por medio de un único mecanismo de instanciación, mediante un único mecanismo de borrado, etc. La base de una única estructura de objetos se relaciona, además, tanto con clases, como con instancias de componentes de software. Esto conlleva, que, durante la modificación de clases, se puedan actualizar inmediatamente todas las instancias en cuanto a datos técnico, por lo que en cada momento se alcanza y/o puede obtenerse una consistencia de datos.

Se continúa sosteniendo, de manera favorable, un coste claramente reducido en el tratamiento de datos, por el hecho de que el árbol de objetos se prevé para la representación de todas las vistas de un componente de software. Todas las vistas del componente de software están presentes, además, en una única estructura, o sea, en el árbol de objetos, de forma que para los procesos de borrado y copia de uno de estos componentes de software sólo sea necesario un único mecanismo de borrado y/o copia.

El sistema presenta un primer tipo de componente de software como clase de un componente de software, lo que permite conseguir una sencilla elaboración de prototipos de un componente de software.

Se puede garantizar un ensayo funcional inmediato de un componente de software, de manera sencilla y sin coste adicional, por ejemplo, mediante compilación, por el hecho de que el primer tipo de componente de software contiene datos y/o funciones ejecutables.

El sistema contiene un algoritmo de generación, previsto para la generación de lenguaje máquina a partir del árbol de objetos (por ejemplo, código S7); lo que posibilita una sencilla elaboración de lenguaje máquina.

Dado que el sistema presenta un segundo tipo de componente de software como instancia del componente de software, se consigue una reutilización de bajo coste de los componentes de software ya producidos. Una de estas instancias de una clase es un duplicado completo de la clase apropiada del componente de software, incluyendo todos los datos y funciones.

Si se modifica la clase apropiada, el sistema presenta medios para la modificación automática de una instancia producida a partir de una clase; lo que garantiza una modificación automática de los datos de una clase de instancias apropiadas.

El primer y el segundo tipo de componente de software tienen el mismo diseño, lo que resulta en un almacenamiento constante de datos tanto para las clases como para las instancias.

Se origina una arquitectura sencilla y clara de los

individuales objetos, por el hecho de que el objeto de un componente de software consiste en nodos objeto.

Se puede obtener un multilingüismo poco costoso del componente de software, dado que un nodo objeto posee un número identificativo para la definición de diferentes textos de lenguas extranjeras de un objeto.

Puede realizarse una elaboración de objetos complejos a partir de objetos individuales más sencillos, por el hecho de que los nodos base pueden jerarquizarse para la generación de objetos más complejos.

Se obtiene una arquitectura básica constante de los objetos y componentes de software, dado que el objeto presenta una función elemental de nodo con un desglose jerárquica.

Adicionalmente se prevé un mecanismo de enlace, que posibilita tanto un enlace de señales como de objetos.

La definición de un determinado comportamiento del objeto puede realizarse de manera segura y sencilla, por el hecho de que el objeto está provisto de una función de nodo base para la determinación de la funcionalidad de un nodo.

Un enlace de objetos más complejos con otros objetos complejos se hace posible, por el hecho de que a los objetos a enlazar se les puede asignar, en cada caso, por lo menos una instancia de una clase común de interfaz. Mediante interconexión de las clases de interfaz se posibilita un intercambio de señales.

La funcionalidad global de un componente de software también para objetos más complejos puede definirse, dado que para la generación de una funcionalidad de un objeto y/o de un componente de software se prevé un mecanismo de enlace para la combinación de señales y/o objetos.

Se garantiza una compatibilidad con las lenguas de especificación habituales en la tecnología de la automatización, dado que el árbol de objetos se prevé para la representación de un diagrama de funciones, de un diagrama de flujos y/o de una secuencia de pasos de un componente de automatización.

También se pueden obtener gráficos en 2D y en 3D de manera sencilla, por el hecho de que el árbol de objetos se prevé para la representación de un gráfico en 2D y/o en 3D.

Una generación de objetos más complejos a partir de los objetos menos complejos generados con anterioridad puede realizarse, de forma que el sistema contenga un concepto tipo/instancia, que se prevé, para que, en caso de modificaciones de una clase, todas las instancias se modifiquen asimismo directamente o de manera controlada.

A continuación se describe la presente invención con mayor detalle y se aclara en base a los ejemplos de ejecución representados en las Figuras.

Muestran:

Fig 1a un diagrama de bloques de la estructura básica de un sistema de automatización con un núcleo OTR como sistema de proyección,

Fig 1b un diagrama de bloques de la estructura básica de un sistema de automatización con un núcleo OTR como sistema de proyección y un núcleo OTR como sistema de ejecución en un controlador programable (=SPS),

Fig 2 una representación esquemática de la estructura básica de un árbol de objetos y su interrelación con las diversas vistas,

Fig 3 una representación esquemática de la interacción en principio de diferentes árboles de objetos,

Fig 4 una representación en principio para la estructura general de datos de un objeto de un árbol de objetos,

Fig 5 una representación en principio para la estructura de datos de un objeto para la elaboración de componentes de software multilingües,

Fig 6 una representación esquemática de un árbol de objetos con objetos desglosados,

Fig 7 una representación esquemática de una ventana de monitor para la definición de las funcionalidades de un árbol de objetos con objetos estructurables jerárquicamente,

Fig 8 una representación ejemplar para la elaboración de diagramas de operaciones mediante un árbol de objetos,

Fig 9 una representación ejemplar para la elaboración de gráficos en 3D por medio de un árbol de objetos,

Fig 10 un árbol de objetos con objetos desglosados empleando el concepto tipo/instancia,

Fig 11 una representación de principio de una estructura de módulos de instalación mediante un concepto tipo/instancia,

Fig 12a una representación ejemplar de una secuencia de pasos,

Fig 12b un árbol de objetos como representación de la secuencia de pasos mostrada en la Fig 12a,

Fig 13 un ejemplo de aplicación práctica del concepto tipo/instancia en la proyección de un sistema de automatización y

fig 14a,b en cada caso, un ejemplo de aplicación práctica del concepto tipo/instancia con tecnología de interfaz OTR.

La Fig 1 presenta una representación de principio de un diagrama de bloques de la proyección y programación de un sistema de automatización AS. El sistema de automatización AS consiste en un sistema de proyección PS, un primer controlador programable SPS1, un segundo controlador programable SPS2, así como un equipo de fabricación FE. El sistema de proyección PS sirve para producir un primer programa de datos DP1, ejecutable sobre el primer controlador programable SPS1, así como para producir un segundo programa de datos DP2, ejecutable sobre el segundo controlador programable SPS2. La programación del primer y segundo programas de datos DP1, DP2 se lleva a cabo con ayuda de un ordenador 1, 2, 3, formado por una unidad computacional 2, un teclado 3 apropiado, así como un monitor 1. En la pantalla 1 del ordenador 1, 2, 3 se encuentra ejemplarmente un árbol de objetos OTR (object tree), que caracteriza la arquitectura fundamental del componente de software de los programas de datos DP1, DP2. La particularidad del ejemplo de ejecución representado en la Fig 1 estriba en que, tanto el primer programa de datos DP1, como también el segundo programa de datos DP2, se basan, en cada caso, en componentes de software, que presentan una estructura de datos de un árbol de objetos OTR con objetos estructurables jerárquicamente.

La estructura de software basada en un árbol de objetos con objetos estructurables jerárquicamente produce mecanismos eficientes de procesamiento, que posibiliten ilustrar todas las estructuras en un modelo uniforme de datos. En conjunto, se origina una proyección uniforme de todas las zonas de la instalación del equipo de fabricación FE con almacenamiento simultáneo, completamente consistente, de datos. Todos los componentes de la instalación del equipo

de fabricación, así como las vistas asignadas, se almacenan dentro de la estructura del árbol de objetos en un único modelo de datos. Una entrada múltiple de datos para diferentes sistemas especiales resulta, por tanto, redundante.

La Fig 1b presenta un diagrama de bloques de la estructura básica de un sistema de automatización con un núcleo OTR como sistema de proyección y un núcleo OTR como sistema de ejecución en un controlador programable (=SPS). Respecto a las relaciones funcionales y a los símbolos de referencia empleados se hace referencia en la Fig. 1.

La Fig 2 presenta una representación esquemática de la estructura básica de un árbol de objetos OTR, de un componente de software K y su interrelación con las diversas vistas SI1..SIx. La base de la estructura del árbol de objetos del componente de software K forma el árbol de objetos OTR, compuesto por objetos estructurables jerárquicamente O1..Om. El componente de software K presenta, por otra parte, una interfaz S, que consiste, en detalle, en interfaces individuales S1..Sy. El componente de software K se almacena en un medio de almacenamiento SP.

La particularidad de la estructura de datos representada en la Fig 2 consiste, entre otros, en que todas las vistas SI..SIx del componente de software K están enlazadas concretamente con datos del árbol de objetos OTR del componente de software K. De este modo se origina un único almacenamiento de datos para todas las vistas, de forma que no sólo se eviten las inconsistencias de los datos, sino más bien que se imposibiliten.

En el ejemplo de ejecución representado en la Fig 2, la primera vista SI1 sirve, por ejemplo, para representar el componente de software en una vista en 3D, la segunda vista SI2 el componente de software K de una vista de diagrama de flujos o de una vista SFC y la tercera vista SI3, por ejemplo, de una vista tabulada del componente de software. La estructura de datos mostrada en la Fig 2 es la base para un mecanismo eficiente de procesamiento, que posibilite ilustrar todas las estructuras, particularmente de una instalación de automatización, en un modelo uniforme de datos. Ya que todos los componentes están sujetos a un y el mismo mecanismo estructural, los componentes de software y sus objetos se pueden enlazar a través de este mecanismo uniforme con otros objetos y componentes de software.

La Fig 3 presenta una representación esquemática de la interacción en principio de varios componentes de software K1..Kn como conexión de diversos árboles de objetos. Los componentes de software K1..Kn presentan además, de nuevo, la misma construcción estructural que el componente de software, explicado ya en conexión con la Fig 2. En el ejemplo de ejecución representado en la Fig 3, los datos del primer componente de software K1 se almacenan en una primera memoria de datos SP1 y los datos del componente de software Kn en una memoria de datos SPy. Mediante flechas discontinuas se sugiere, sin embargo, que los componentes de software K1..Kn pueden almacenarse también, en cada caso, en otras memorias de datos.

La Fig 4 presenta una representación en principio de la estructura general de datos de un objeto O1..Om, tal y como se emplea dentro de un árbol de objetos (compare con la Fig 2). El objeto O1..Om mostrado en la Fig 4 consiste en un bloque de datos KF, que describe la función de nodo del objeto O1..Om. La función elemental de nodo del objeto O1..Om consiste, además, en un desglose jerárquico por medio de indicadores Z1..Z4, así como en una definición de la función de nodo base BKF. La función de nodo base BKF define, además, el comportamiento de objetos del objeto O1..Om. La funcionalidad puede ser, por ejemplo, una función AND, que presenta la función de enlazar los nodos objeto subalmacenados a través de una función lógica AND. Los nodos base se pueden jerarquizar y se pueden utilizar, por tanto, para la generación de objetos más complejos. El indicador Z2 remite a un llamado objeto Master M, mientras que el indicador Z1 remite al primer objeto hijo FC (First Child). De manera similar, en el bloque de datos KF, que define la función de nodo del objeto O1..Om, se encuentran contenidas otras funciones de nodo KN1..KN5, sirviendo el indicador Z4 para el desglose de la función de nodo base BKF.

La Fig 5 presenta una representación en principio para la estructura de datos de un objeto para la elaboración de componentes de software multilingües. La construcción estructural del objeto corresponde además a la estructura básica ya mostrada en conexión con la Fig 4. Adicionalmente a las funciones de nodo mostradas en la Fig 4, el objeto representado en la Fig 5 contiene otra función de nodo KN5, que remite, a través de un indicador Z5, a una tabla de lenguas extranjeras. En la tabla de lenguas extranjeras se enlazan, en cada caso, a través de identificaciones diferentes idiomas y ediciones de texto para las vistas asignadas al respectivo objeto. El concepto uniforme de retención de datos así disponible posibilita, por tanto, de manera sencilla también un procedimiento uniforme para la traducción de todos los componentes de la instalación a diferentes lenguas extranjeras, sin que sean necesarias modificaciones en la construcción estructural básica del propio componente de software.

La Fig 6 presenta una representación esquemática de un árbol de objetos OTR. El árbol de objetos OTR consiste en objetos O1..O13. La funcionalidad del árbol de objetos OTR representado ejemplarmente en la Fig 6 produce señales definidas a través de un mecanismo de enlace SL para la combinación de objetos individuales y/o por medio de objetos. El objeto O1 representa un componente, el objeto O2 una interfaz, el objeto O3 define datos, el objeto O4 funciones. Los objetos O5, O6 ilustran una entrada y/o una salida, mientras que los objetos O12 y O13 señalan la señal de entrada A y/o señal de salida B apropiada en cada caso. El objeto 07 caracteriza una señal DATA y el objeto una señal P, enlazada como señal B a través de un indicador ZA con el objeto O13, es decir, con la señal B. El objeto O9 forma una puerta Y, mientras que los objetos O10 y O11 caracterizan la señal A, que se suministra como enlace a través de un indicador ZB. El objeto O7 se enlaza con el objeto O11 a través de un indicador ZC.

La Fig 6 presenta un flujo se señales resultante del mecanismo de enlace SL para el enlace de las señales "señal A" del objeto O12 y del objeto O7 ("DATA") a través de una puerta Y O9. El resultado de este enlace se suministra de nuevo a la señal B del objeto O8 a través de un enlace ZA. Con ayuda del mecanismo de enlace mostrado ejemplarmente en la Fig 6 se pueden enlazar también unos objetos complejos con otros, como por ejemplo, el intercalado dinámico de una máquina en la vista de una instalación.

La Fig 7 presenta una representación esquemática de una ventana de monitor 1 para la definición de las funcionalidades de un árbol de objetos OTR con objetos estructurables jerárquicamente. La ventana de monitor se subdivide además, en principio, en tres subzonas del monitor B1, B2, B3. La primera subzona del monitor B1 presenta el árbol de objetos OTR a proyectar con los objetos individuales, mientras que la segunda zona del monitor B2 muestra en una tabla las funciones de nodo de los objetos base. Un gráfico asignado a estos objetos base se representa gráficamente en la tercera zona del monitor B3 -si existe-. Se lleva a cabo una asignación de los respectivos objetos base a los objetos individuales a través del desglose de los objetos individuales del árbol de objetos OTR de la primera subzona del monitor con los respectivos objetos base de la segunda zona del monitor B2.

La Fig 8 presenta una representación ejemplar para la elaboración de diagramas de operaciones mediante un árbol de objetos. Como ejemplo, en la Fig 8 se muestra una puerta lógica, compuesta en cada caso por nodos base.

La Fig 9 presenta una representación ejemplar para la elaboración de gráficos en 3D mediante un árbol de objetos. Los respectivos gráficos en 3D para un eje lineal consisten en subobjetos apropiados, como guía y accionamiento. Así, el gráfico en 3D del accionamiento representado en la Fig 9 está compuesto, por ejemplo, por un llamado resolvedor, así como por un motor. La estructura completa de árbol del gráfico en 3D se deposita además en la estructura base del árbol de objetos.

La Fig 10 presenta un árbol de objetos con objetos desglosados empleando el concepto tipo/instancia. El concepto tipo/instancia posibilita la generación de nuevos objetos más complejos a partir de objetos menos complejos generados con anterioridad. Además, se produce un sistema jerárquico de instanciación, que se superpone a la jerarquía estructural de la instalación de automatización. Si en una clase se modifica, se añade, se desplaza o se borra, una propiedad, se modifican también directamente todas las instancias, bien automáticamente o bien de manera controlada por el usuario, de forma que se conserve la consistencia de la estructura de datos. La particularidad del concepto tipo/instancia representado en la Fig 10 consiste en, que también las clases poseen ya funcionalidad total frente a los habituales conceptos orientados al objeto. Esta capacidad de funcionalidad total se basa en, que también las clases poseen ya datos con memoria reservada y módulos ejecutables de programa. La ventaja de uno de estos procedimientos consiste en que las clases pueden tratarse como prototipos, que pueden depurarse inmediatamente, sin instanciación adicional, es decir, sin coste adicional. La funcionalidad total de las clases contiene de nuevo todas las vistas y facetas de la tecnología de automatización.

La Fig 11 presenta una representación de principio de este tipo para una estructura de módulos de instalación por medio de un concepto tipo/instancia.

La Fig 12a presenta una representación ejemplar de una cadena de pasos, tal y como se emplea, por ejemplo, en sistemas de proyección habituales de la tecnología de la automatización. Una cadena de pasos de este tipo consiste en acciones y transiciones, es decir, estados y transiciones de estado. Frente a esto, en la Fig 12b se presenta un árbol de objetos como representación de la cadena de pasos mostrada en la Fig 12a.

La Fig 13 presenta un ejemplo para la aplicación práctica del concepto tipo/instancia en la proyección de un sistema de automatización.

Las Fig 14a,b presentan un ejemplo para la aplicación práctica del concepto tipo/instancia con tecnología de interfaz OTR.

REIVINDICACIONES

1. Componente de software (K1..Kn) de un sistema de automatización con una estructura de datos constituida por un árbol de objetos (OTR) con objetos estructurables jerárquicamente (O1..Om), previéndose el árbol de objetos (OTR) para la representación de vistas del componente de software (K1..Kn), estando todas las vistas del componente de software (K1..Kn) enlazadas concretamente con datos del árbol de objetos (OTR) del componente de software, de forma que resulte un único almacenamiento de datos para todas las vistas, sirviendo las vistas para la representación del componente de software en una vista en 3D y vista de diagrama de flujos, particularmente vista SFC.

2. Componente de software acorde a la Reivindicación 1, **caracterizado** porque el componente de software (K1..Kn) presenta un primer tipo de componente de software (KA1) como clase (Kl).

3. Componente de software según al menos una de las Reivindicaciones 1 a 2, **caracterizado** porque el primer tipo de componente de software (KA1) contiene datos y/o funciones ejecutables.

4. Componente de software según al menos una de las Reivindicaciones 1 a 3, **caracterizado** porque el componente de software (K1..Kn) presenta un segundo tipo de componente de software (KA2) como instancia (En) del componente de software.

5. Componente de software según al menos una de las Reivindicaciones 1 a 4, **caracterizado** porque se prevén medios para la modificación automática de una instancia producida a partir de una clase, si se modifica la clase correspondiente.

6. Sistema según al menos una de las Reivindicaciones 1 a 5, **caracterizado** porque el primer (KA1) y el segundo (KA2) tipo de componente de software tienen el mismo diseño.

7. Componente de software (K1..Kn) según al menos una de las Reivindicaciones 1 a 6, **caracterizado** porque un objeto (O1..Om) de un componente de software presenta por lo menos un nodo objeto.

8. Componente de software (K1..Kn) según al menos una de las Reivindicaciones 1 a 7, **caracterizado** porque un nodo objeto individual es un objeto.

9. Componente de software (K1..Kn) según al menos una de las Reivindicaciones 1 a 8, **caracterizado** porque el nodo objeto consiste en indicadores a otros nodos y/o en números identificativos de otros nodos. (Identificadores)

10. Componente de software (K1..Kn) según al menos una de las Reivindicaciones 1 a 9, **caracterizado** porque un nodo objeto (Ks) posee un número identificativo para la definición de diferentes lenguas extranjeras de un objeto (O1..Om).

11. Componente de software (K1..Kn) según al menos una de las Reivindicaciones 1 a 10, **caracterizado** porque los nodos base (BKN) pueden jerarquizarse para la generación de objetos más complejos.

12. Componente de software (K1..Kn) según al menos una de las Reivindicaciones 1 a 11, **caracterizado** porque el objeto (O1..On) presenta una función elemental de nodo (KF), particularmente con un desglose jerárquico.

13. Componente de software (K1..Kn) según al menos una de las Reivindicaciones 1 a 12, **caracterizado** porque el objeto (O1..On) está provisto de una función de nodo base (BKF) para la determinación de la funcionalidad de un nodo y/o de nodos subordinados.

14. Componente de software (K1..Kn) según al menos una de las Reivindicaciones 1 a 13, **caracterizado** porque para la generación de una funcionalidad de un objeto y/o de un componente de software se prevé un mecanismo de enlace para la combinación de señales y/o objetos.

15. Componente de software (K1..Kn) según al menos una de las Reivindicaciones 1 a 14, **caracterizado** porque el árbol de objetos (OTR) se prevé para la realización de un diagrama de funciones, de un diagrama de flujo y/o de una secuencia de programa de un componente de automatización.

16. Componente de software (K1..Kn) según al menos una de las Reivindicaciones 1 a 15, **caracterizado** porque el árbol de objetos (OTR) para construir un gráfico en 2D y/o en 3D.

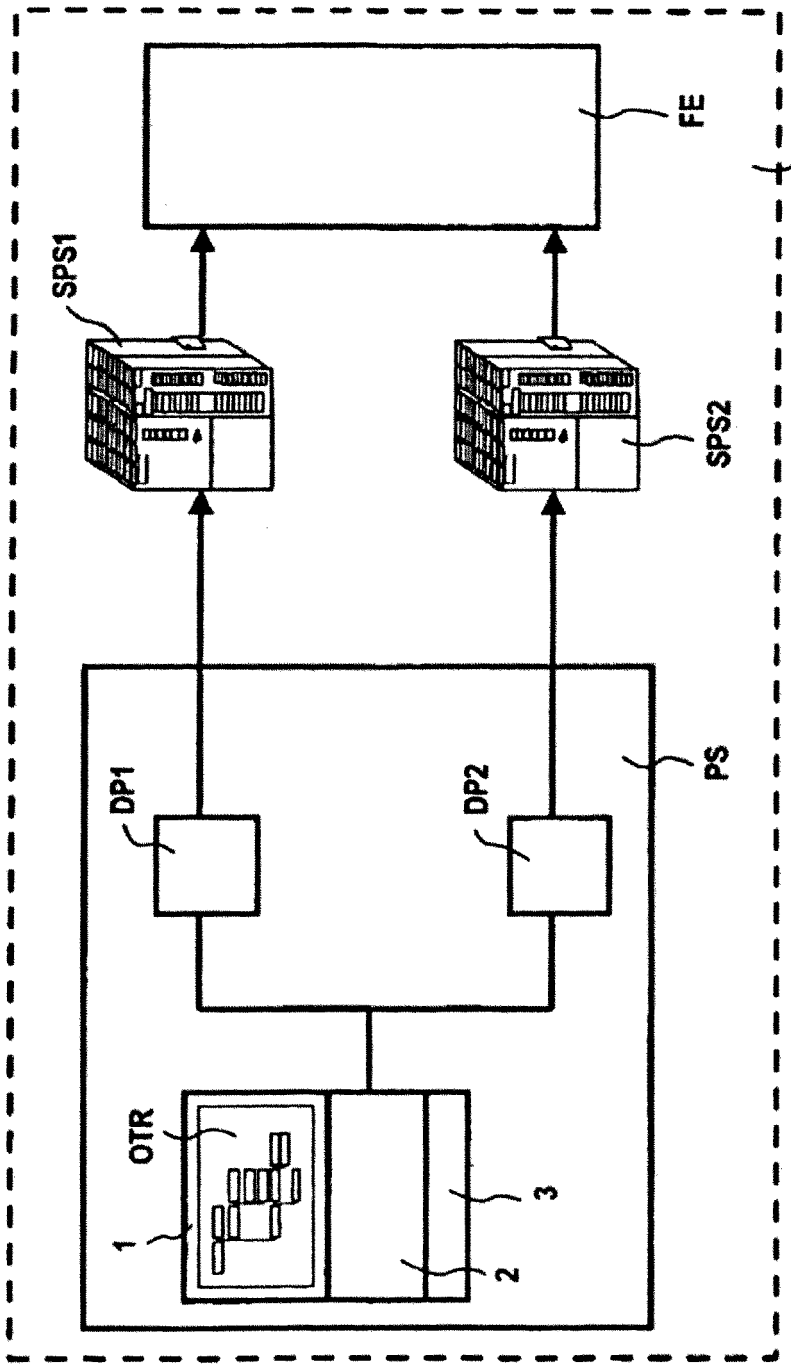

FIG 1a

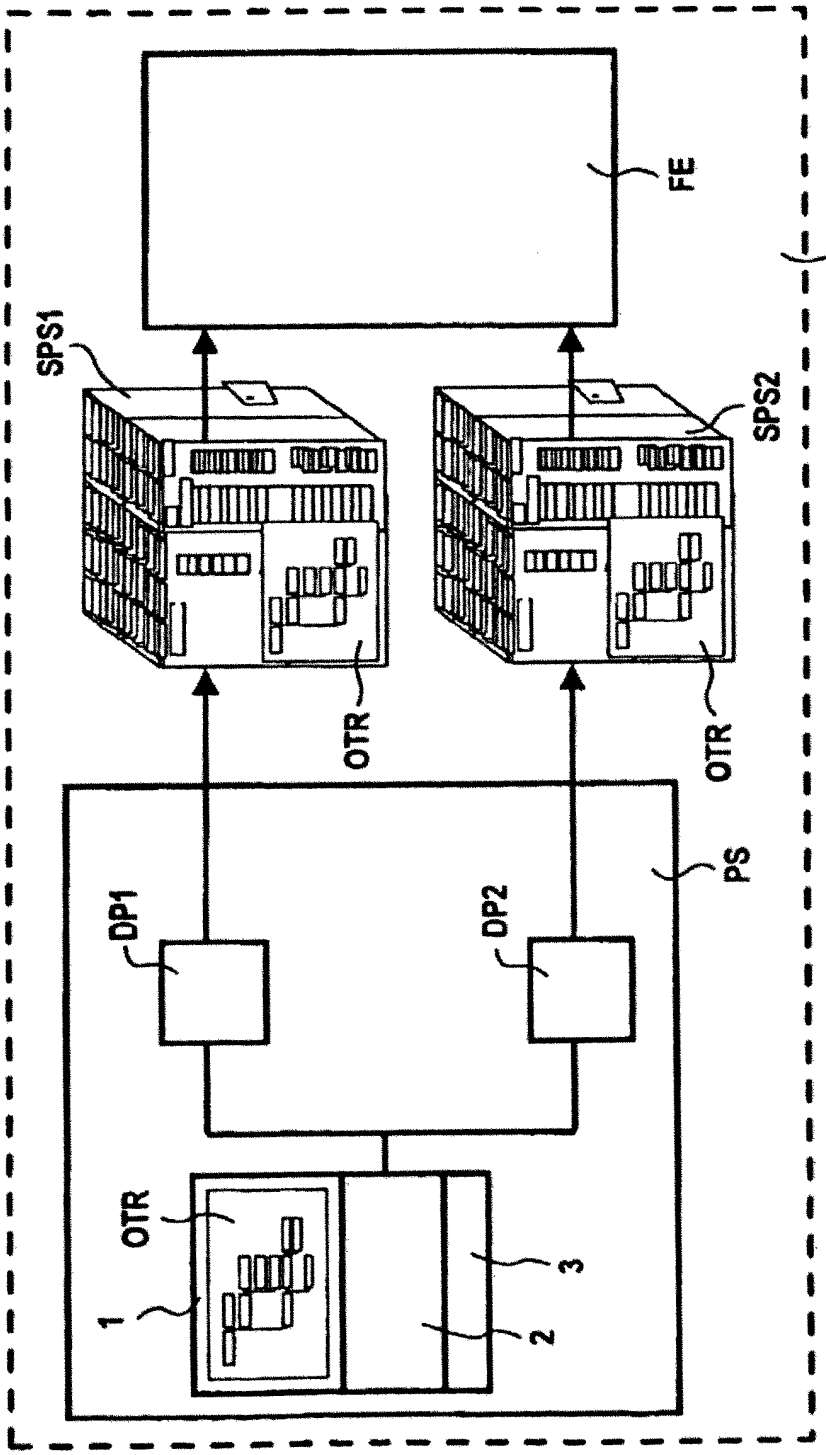

FIG 1b

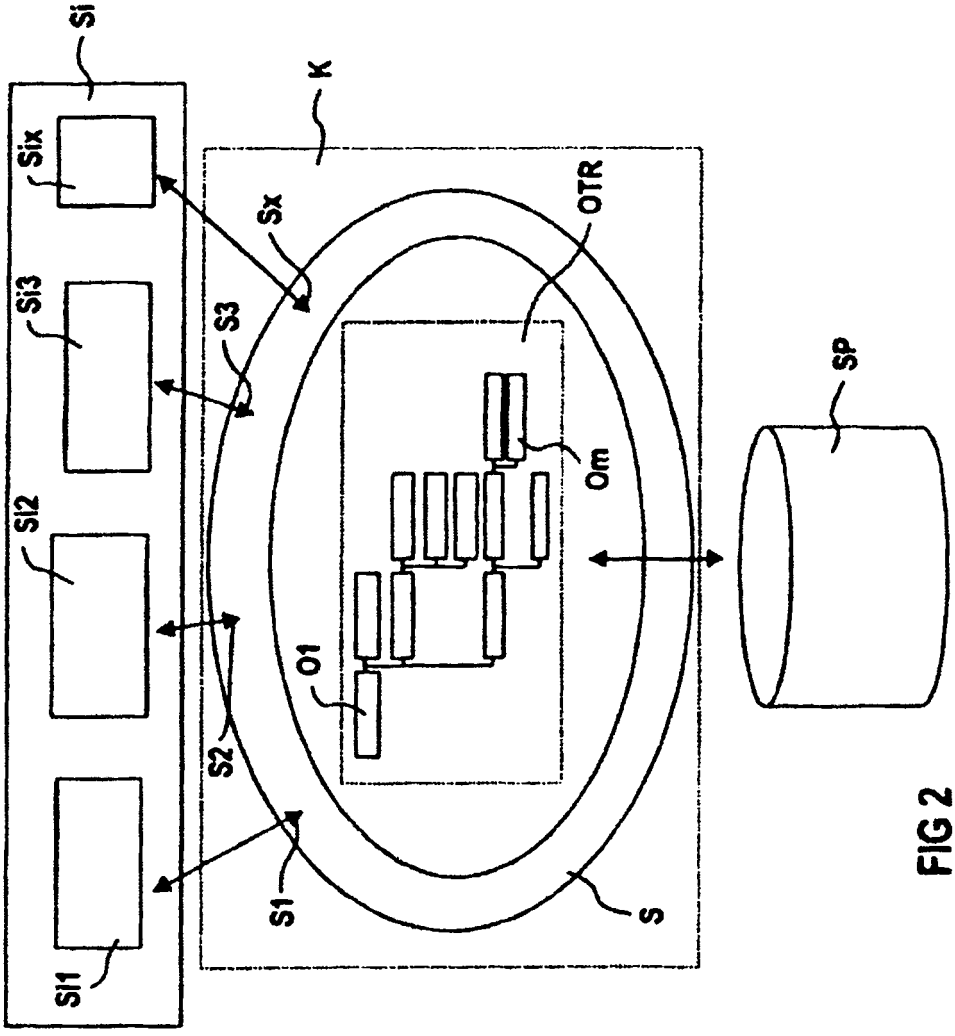

FIG 2

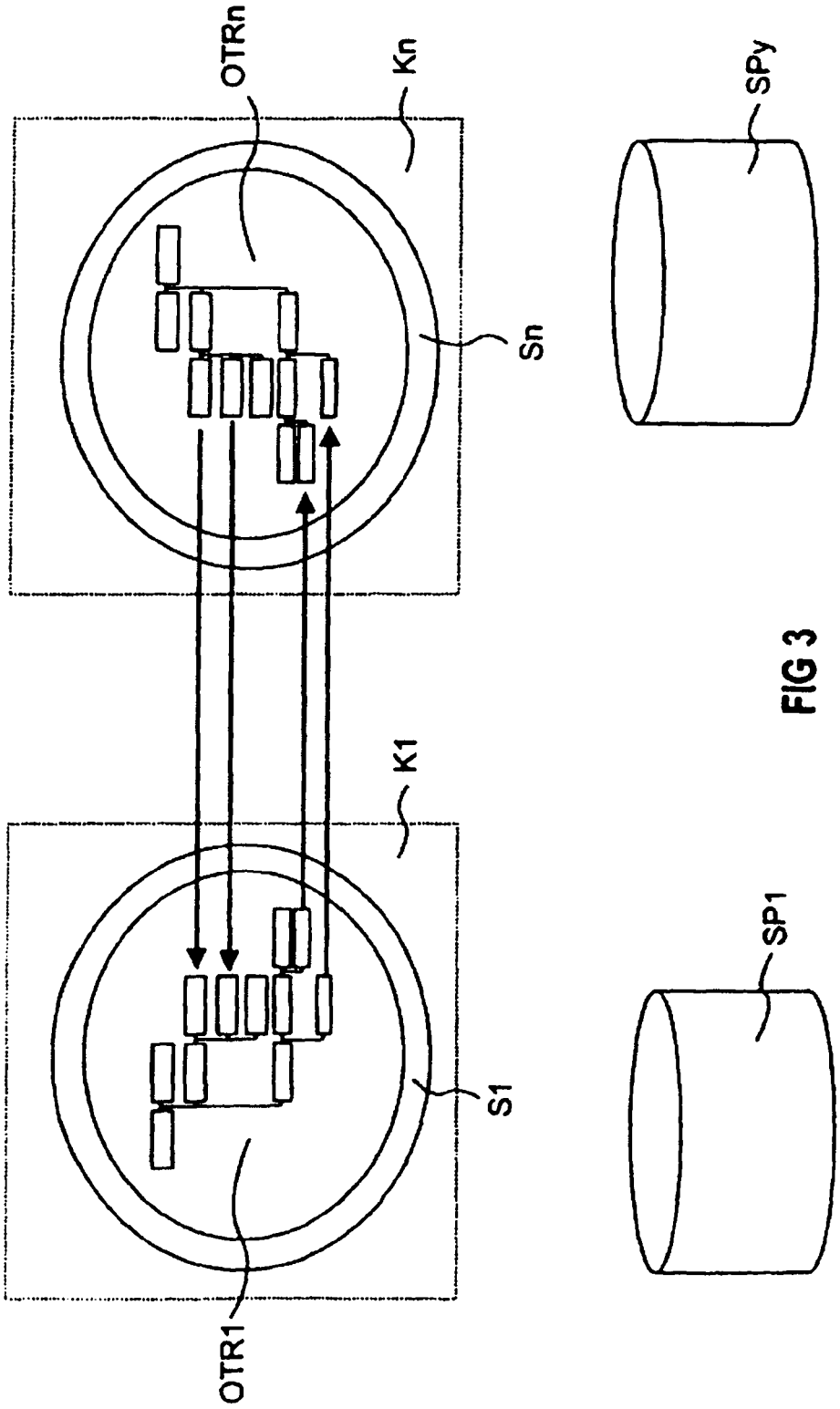

FIG 3

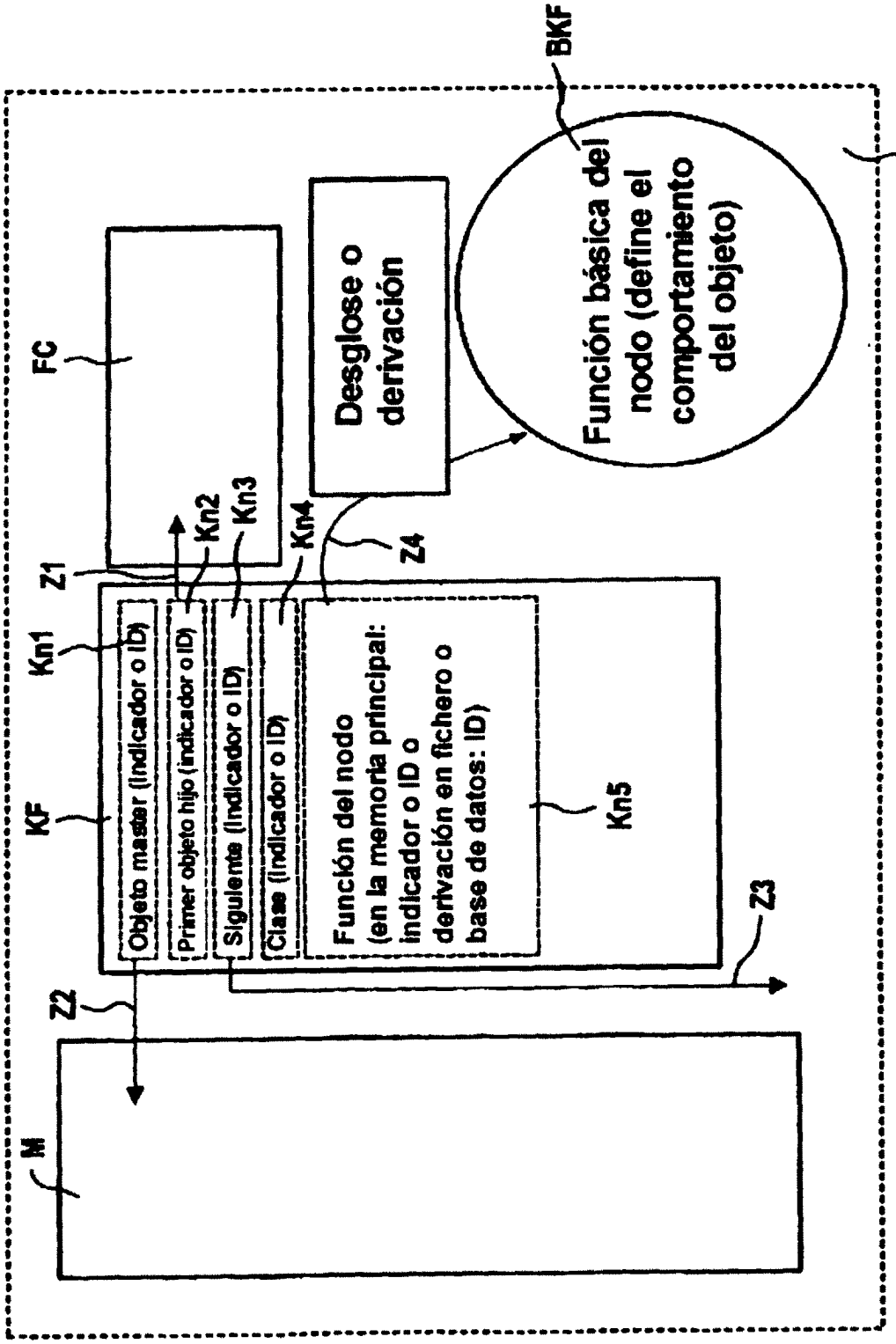
BKF
Función básica del nodo (define el comportamiento del objeto)
01.. 0m
Desglose o derivación
FC
Kn2
Kn3
Kn4
Z4
Z1
Kn1
KF
Objeto master (indicador o ID)
Primer objeto hijo (indicador o ID)
Siguiente (indicador o ID)
Clase (indicador o ID)
Función del nodo (en la memoria principal: indicador o ID o derivación en fichero o base de datos: ID)
Kn5
Z3
Z2
M

FIG 4

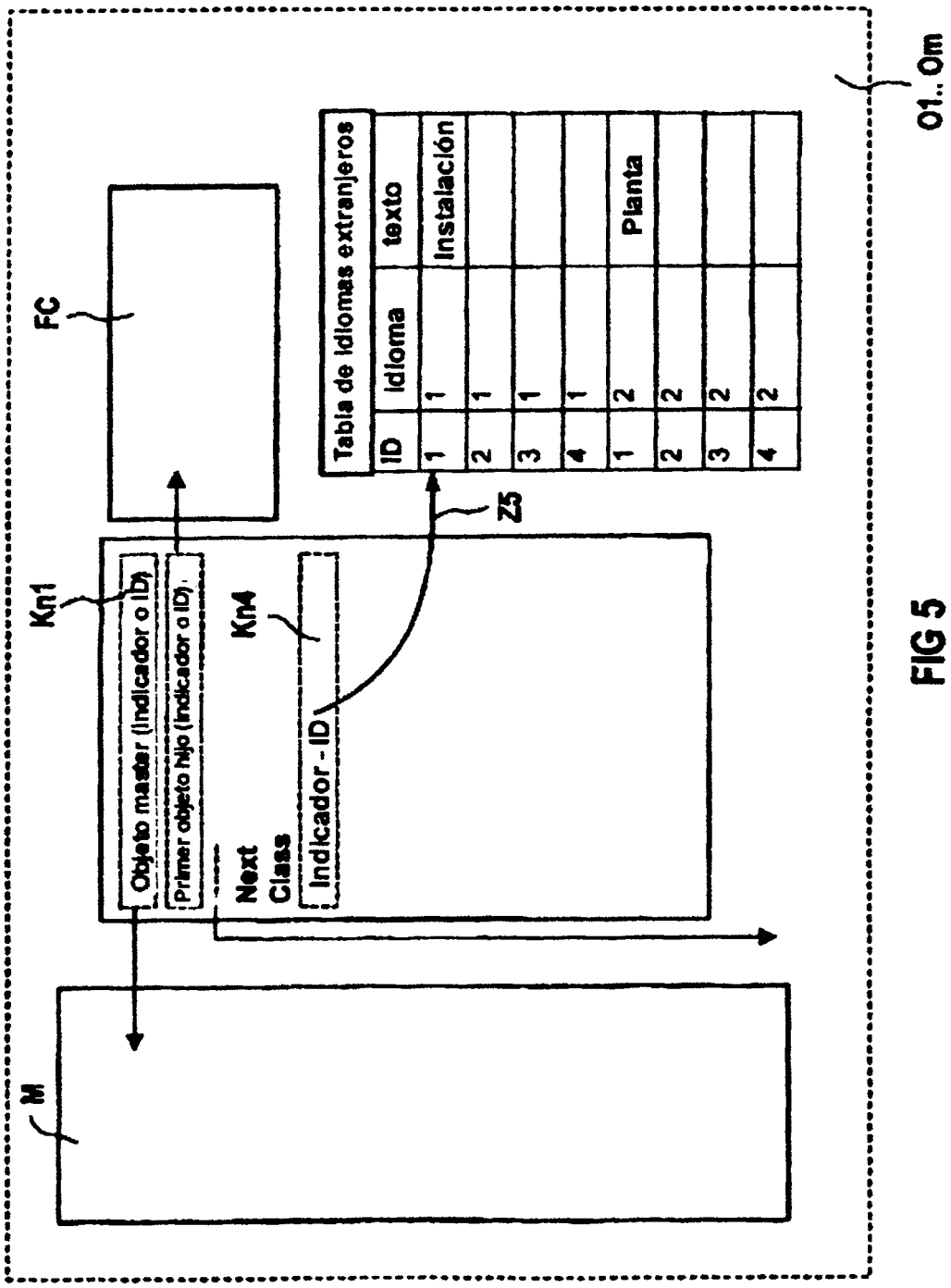
01..0m
FC
Tabla de idiomas extranjeros
ID
idioma
texto
1
1
Instalación
2
1
3
1
4
1
1
2
Planta
2
2
3
2
4
2
Z5
Kn1
Kn4
Objeto master (Indicador o ID)
Primer objeto hijo (Indicador o ID)
Next
Class
Indicador - ID
M

FIG 5

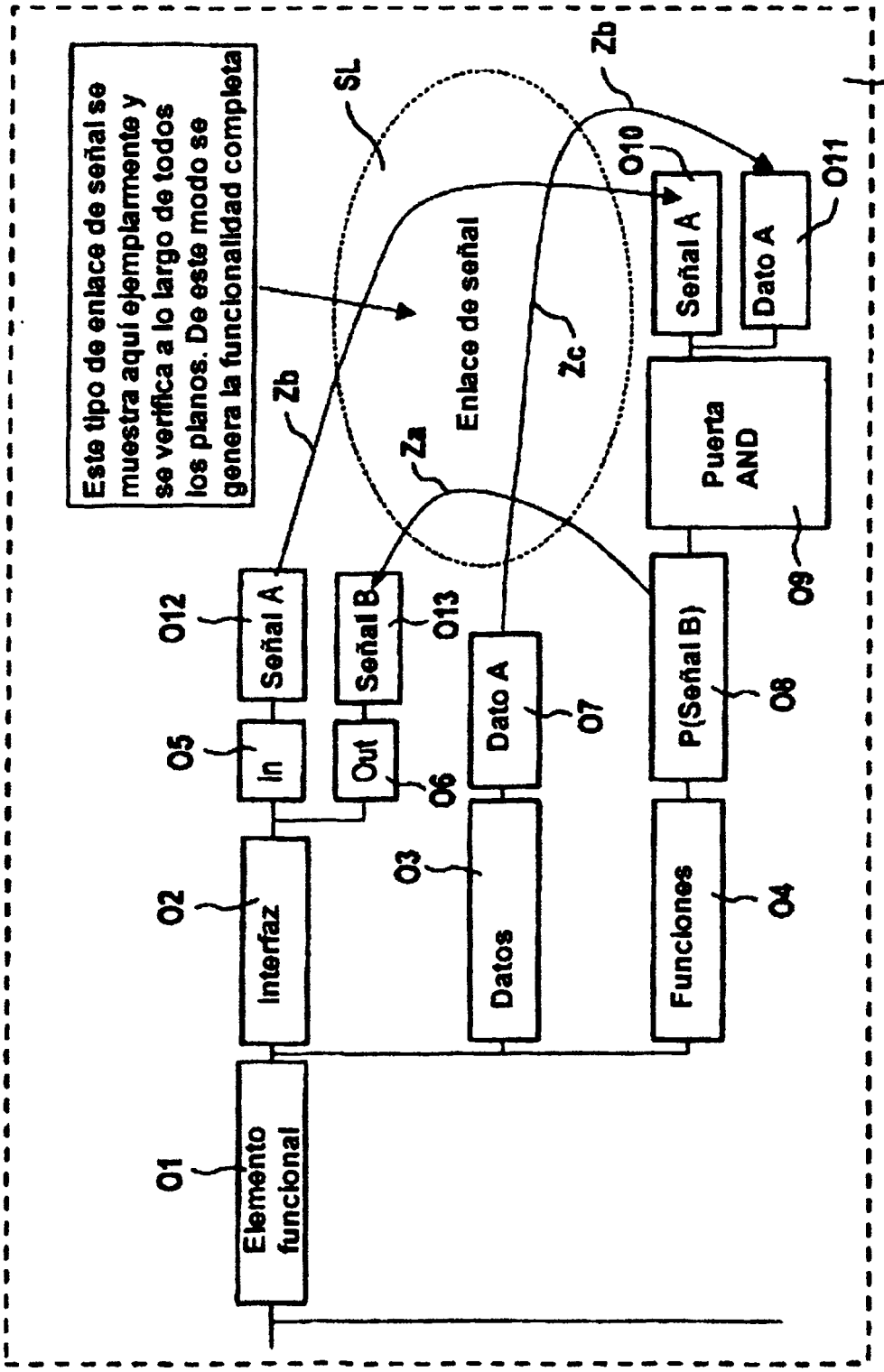
Este tipo de enlace de señal se muestra aquí ejemplarmente y se verifica a lo largo de todos los planos. De este modo se genera la funcionalidad completa
SL
Zb
Zb
OTR
O10
O11
Señal A
Dato A
Enlace de señal
Zc
Za
Puerta AND
Zb
Za
O12
O13
Señal A
Señal B
O5
In
Out
O6
Dato A
O7
P(Señal B)
O8
O9
O2
Interfaz
O3
Datos
Funciones
O4
O1
Elemento funcional

FIG 6

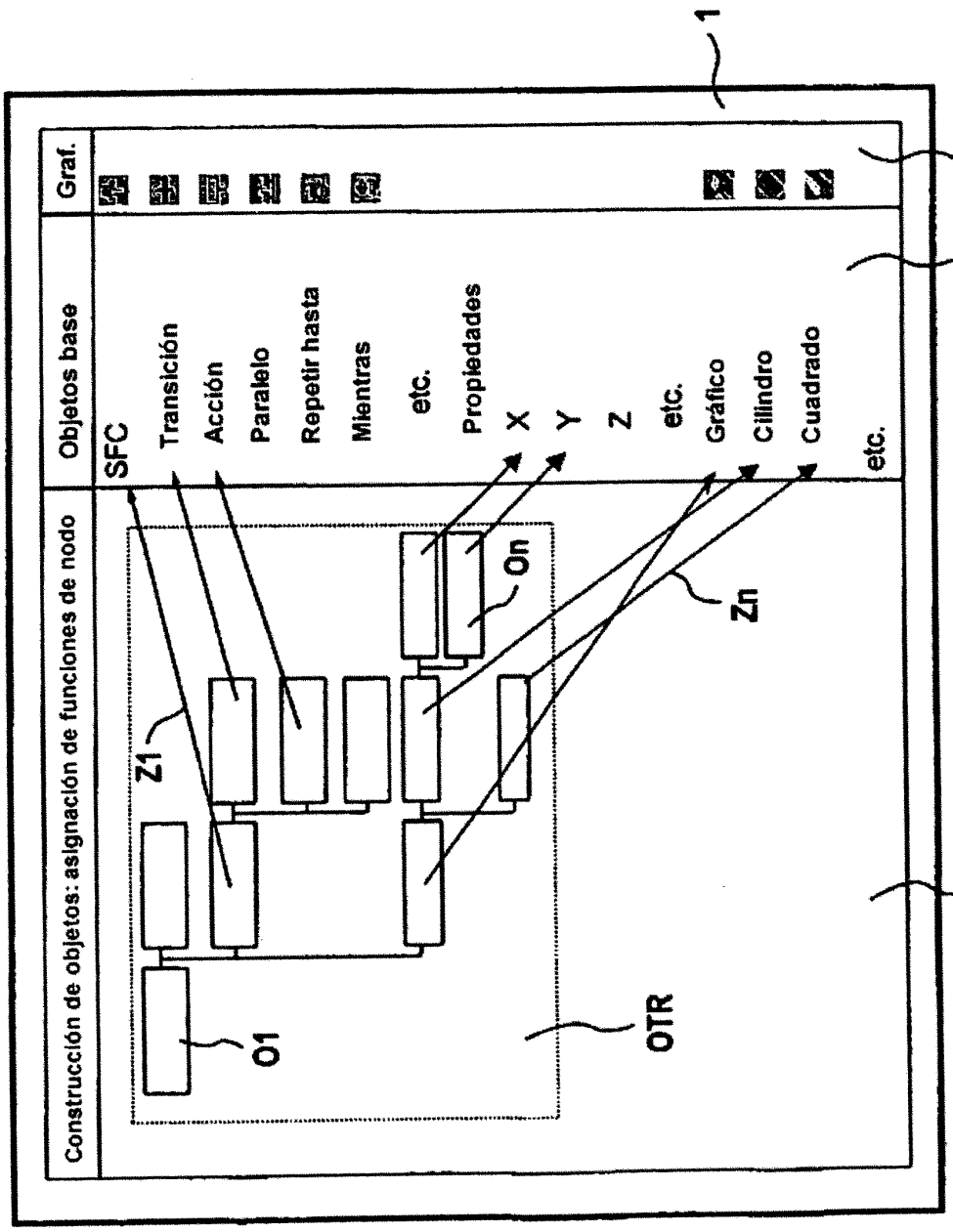
1
Graf.
Objetos base
Construcción de objetos: asignación de funciones de nodo
SFC
Transición
Acción
Paralelo
Repetir hasta
Mientras
etc.
Propiedades
X
Y
Z
etc.
Gráfico
Cilindro
Cuadrado
etc.
B3
B2
B1
Z1
On
Zn
O1
OTR

FIG 7

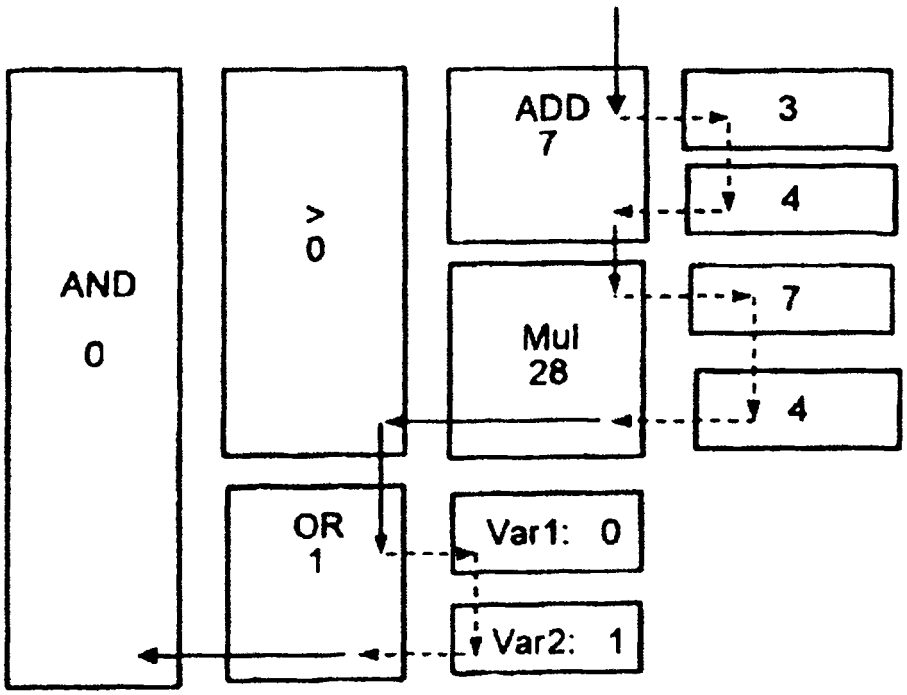

FIG 8

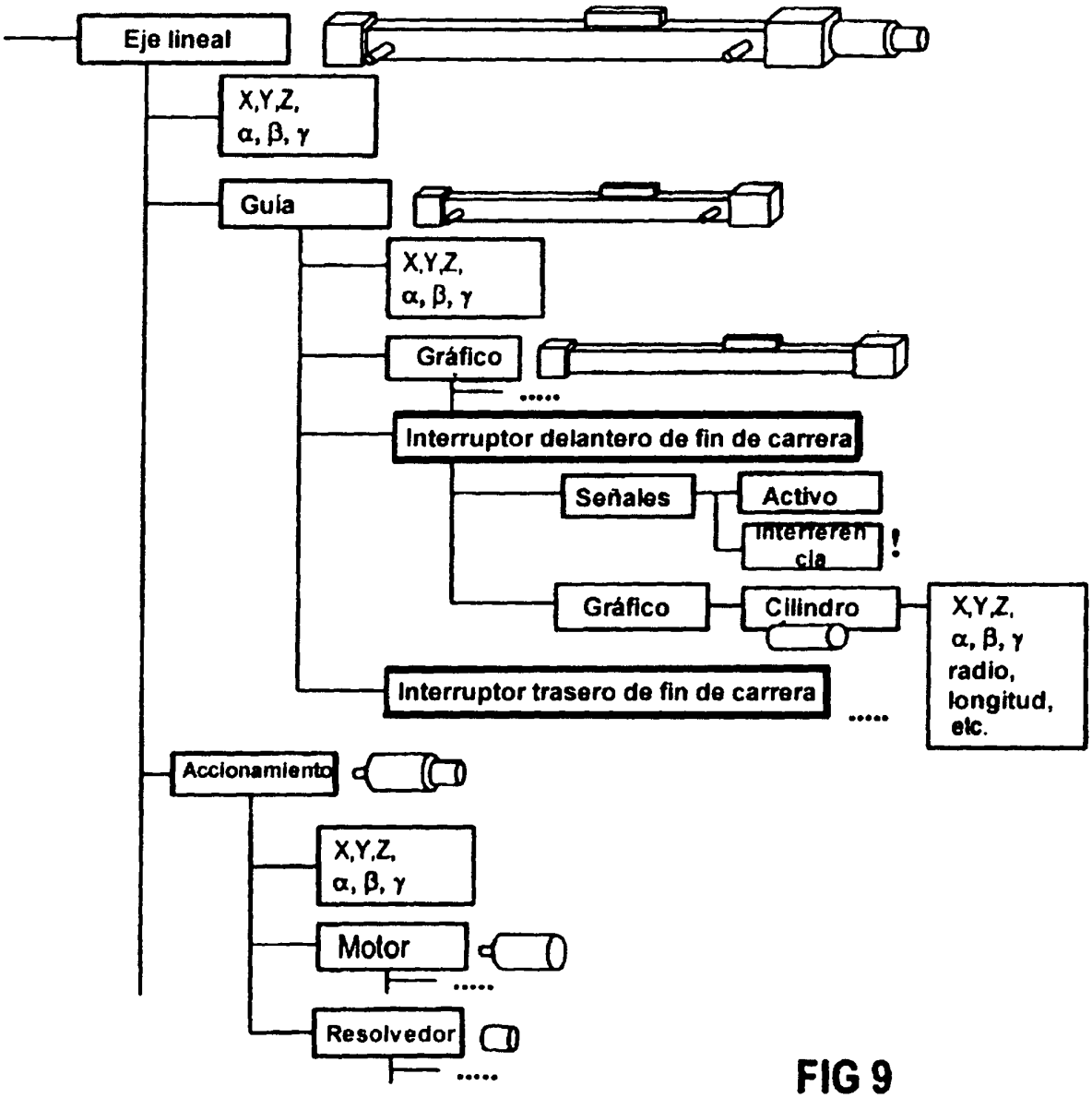
Eje lineal
X,Y,Z,
α, β, γ
Guía
X,Y,Z,
α, β, γ
Gráfico
.....
Interruptor delantero de fin de carrera
Señales
Activo
Interferencia
!
Gráfico
Cilindro
X,Y,Z,
α, β, γ
radio,
longitud,
etc.
Interruptor trasero de fin de carrera
.....
Accionamiento
X,Y,Z,
α, β, γ
Motor
.....
Resolvedor
.....

FIG 9

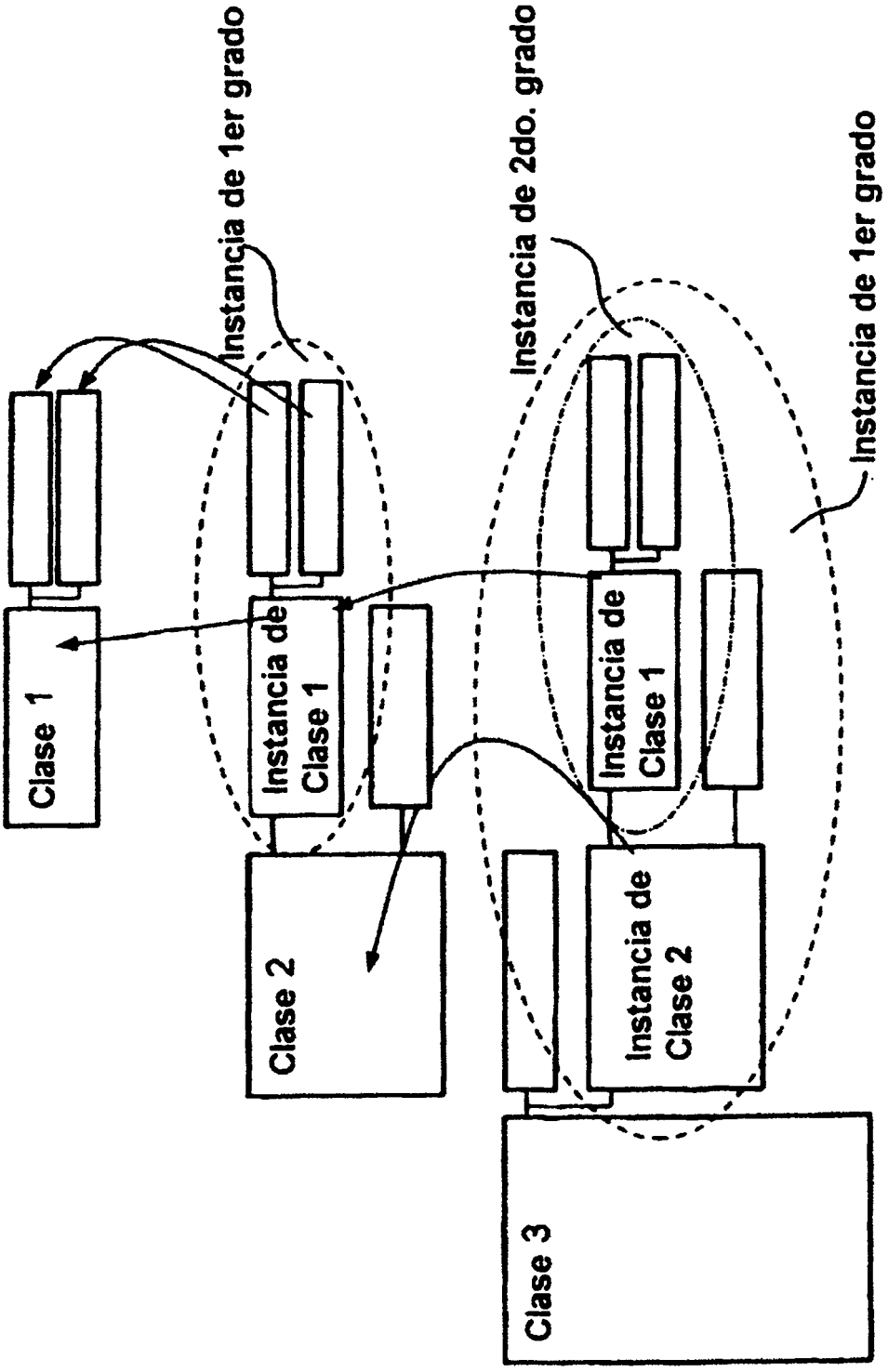
Instancia de 1er grado
Instancia de 2do. grado
Instancia de 1er grado
Clase 1
Instancia de Clase 1
Instancia de Clase 1
Clase 2
Instancia de Clase 2
Clase 3

FIG 10

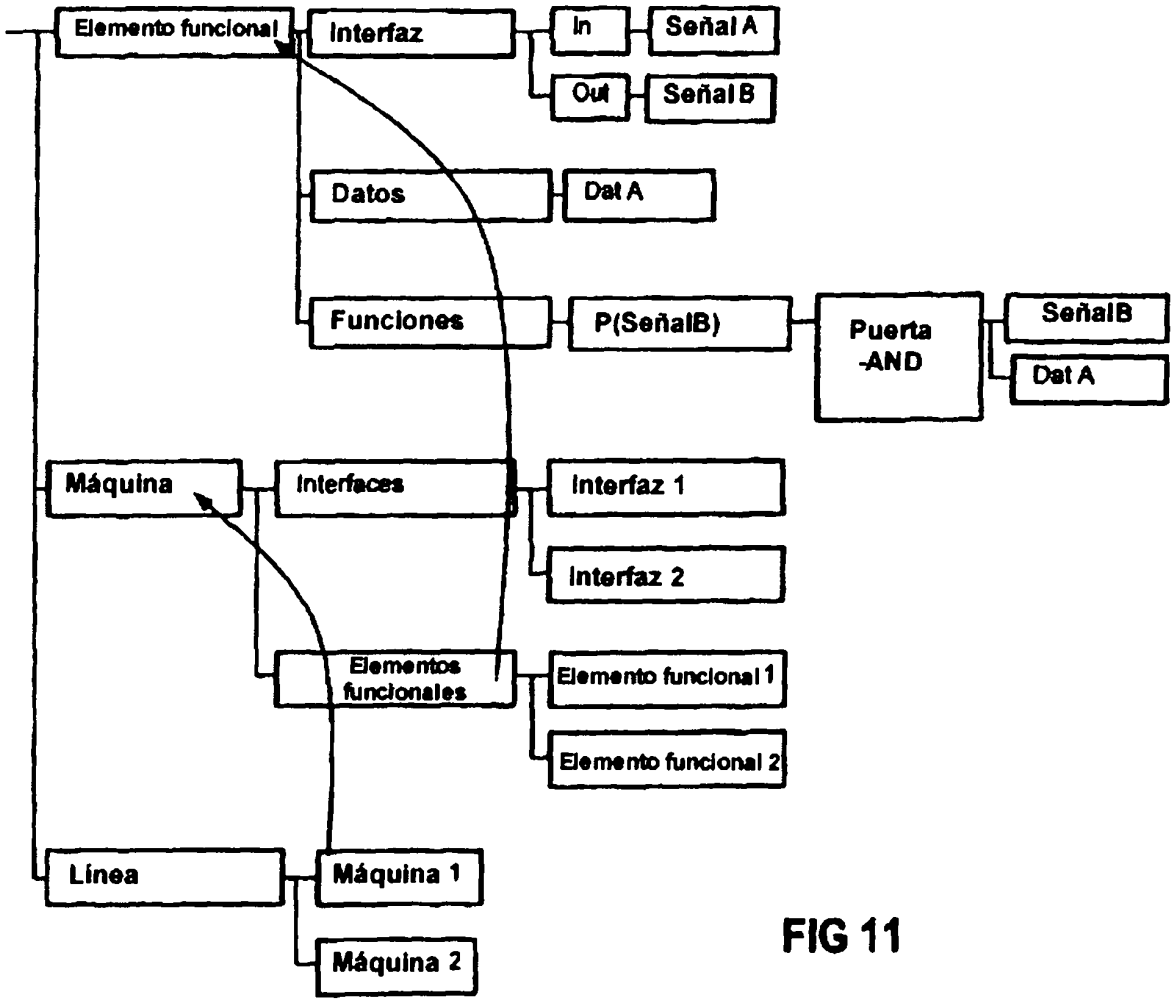
Elemento funcional
Interfaz
In
Señal A
Out
Señal B
Datos
Dat A
Funciones
P(SeñalB)
Puerta -AND
SeñalB
Dat A
Máquina
Interfaces
Interfaz 1
Interfaz 2
Elementos funcionales
Elemento funcional 1
Elemento funcional 2
Línea
Máquina 1
Máquina 2

FIG 11

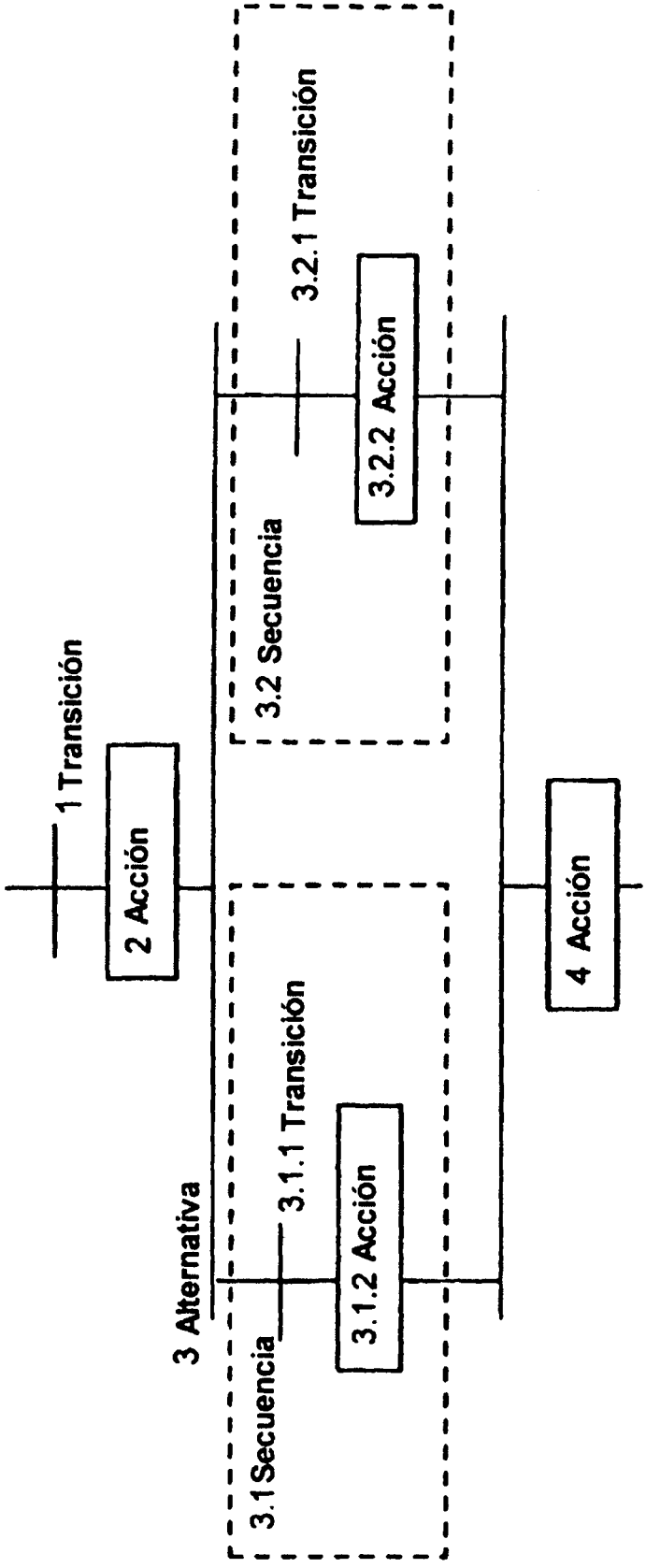
1 Transición
2 Acción
3 Alternativa
3.1 Secuencia
3.1.1 Transición
3.1.2 Acción
3.2 Secuencia
3.2.1 Transición
3.2.2 Acción
4 Acción

FIG 12a

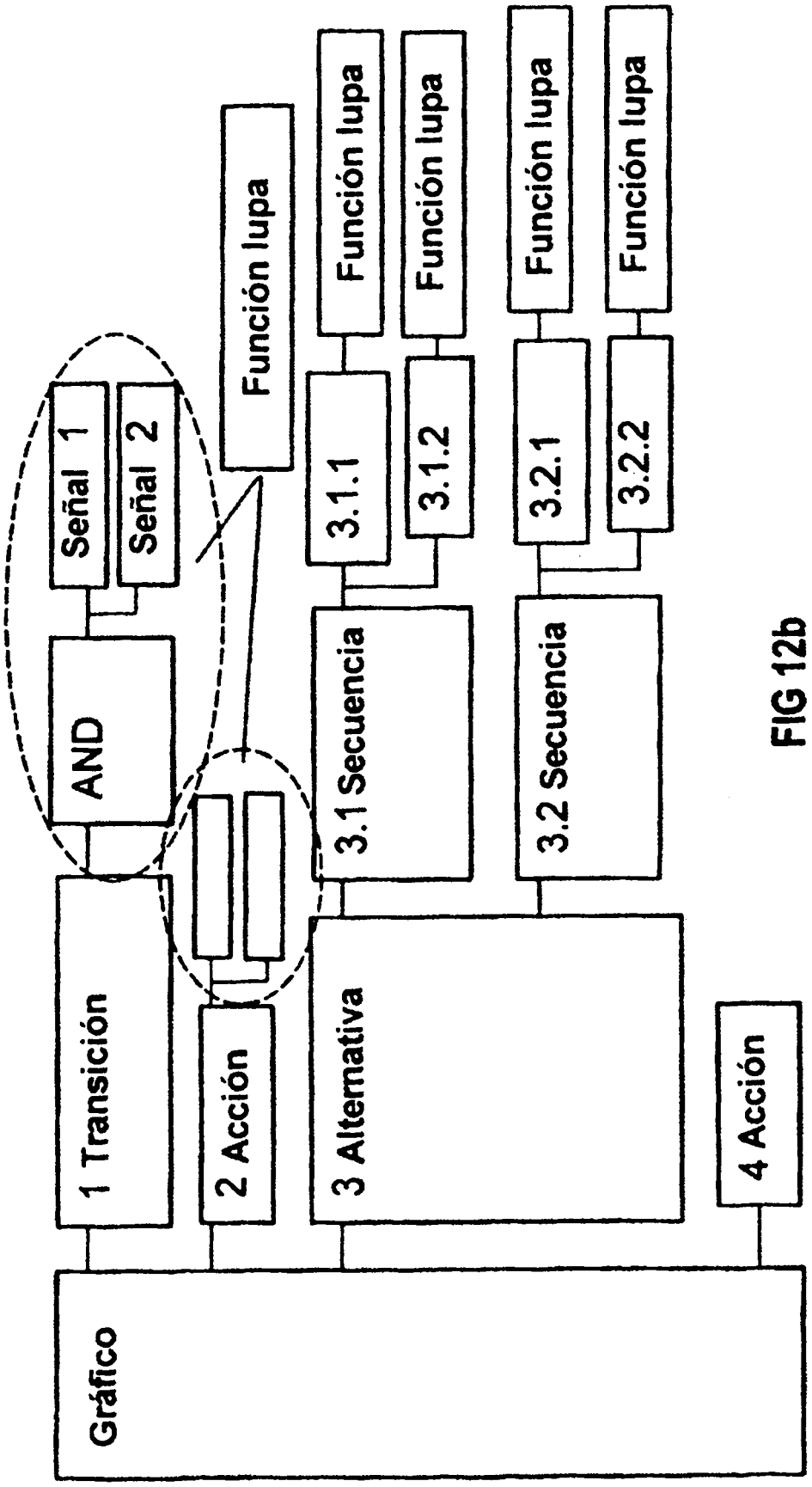
Gráfico
1 Transición
Señal 1
Señal 2
AND
2 Acción
3 Alternativa
3.1 Secuencia
3.2 Secuencia
4 Acción
Función lupa
Función lupa
Función lupa
Función lupa
Función lupa
3.1.1
3.1.2
3.2.1
3.2.2

FIG 12b

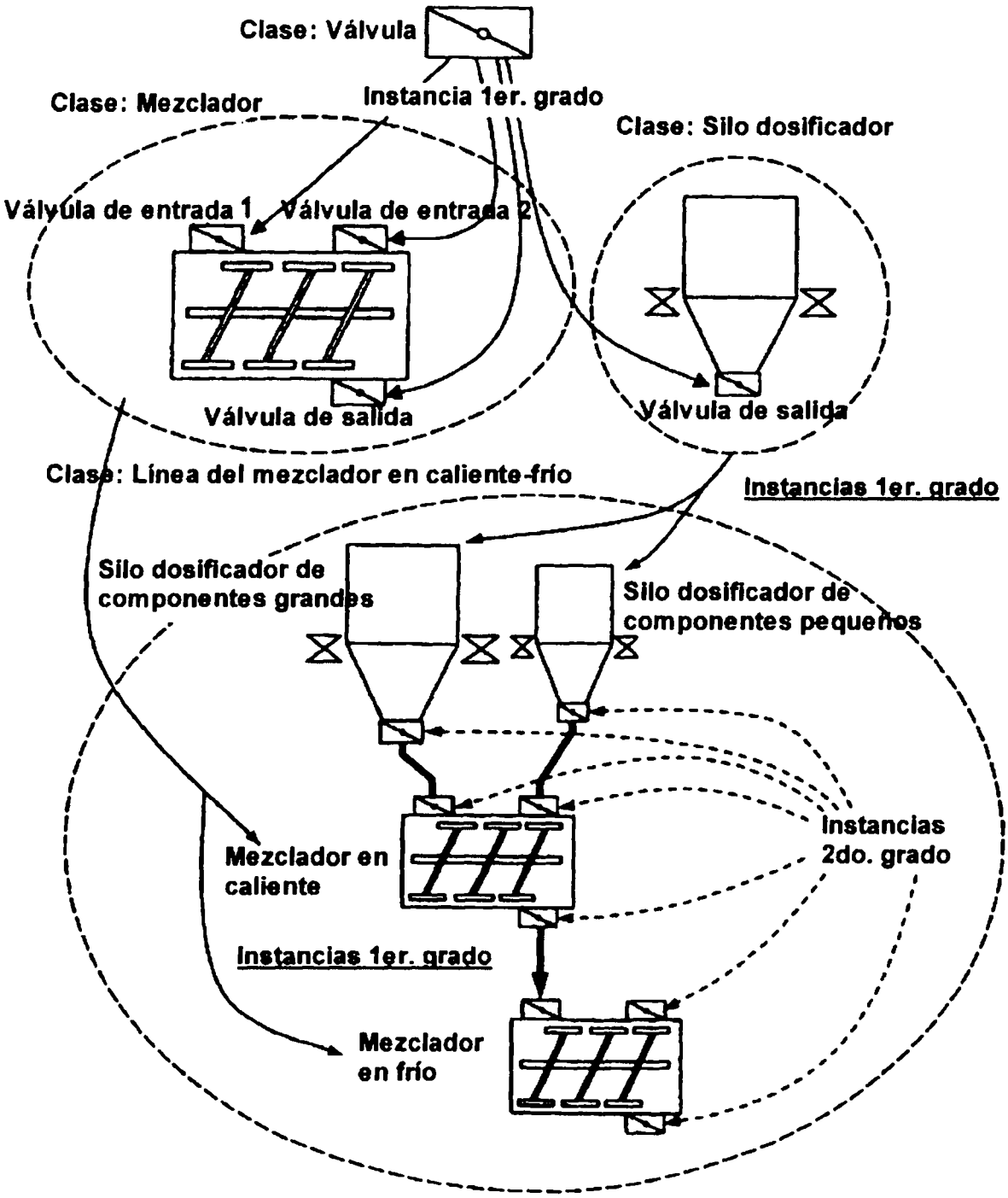
Clase: Válvula
Instancia 1er. grado
Clase: Mezclador
Clase: Silo dosificador
Válvula de entrada 1
Válvula de entrada 2
Válvula de salida
Válvula de salida
Clase: Línea del mezclador en caliente-frío
Instancias 1er. grado
Silo dosificador de componentes grandes
Silo dosificador de componentes pequeños
Instancias 2do. grado
Mezclador en caliente
Instancias 1er. grado
Mezclador en frío

FIG 13

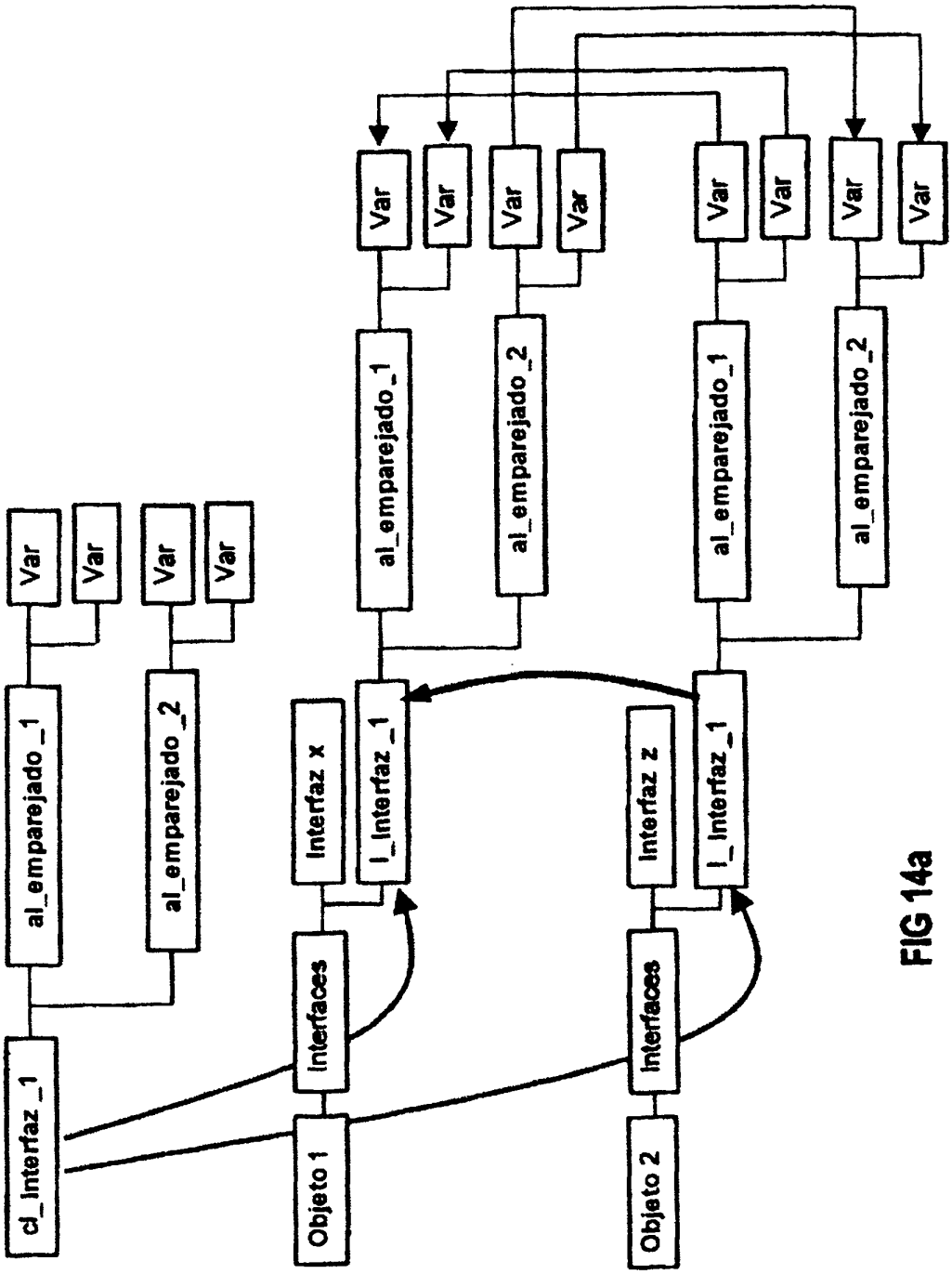
cl_interfaz_1
al_emparejado_1
al_emparejado_2
Var
Var
Var
Var
Interfaces
Objeto 1
Interfaz x
I_interfaz_1
al_emparejado_1
al_emparejado_2
Var
Var
Var
Var
Interfaces
Objeto 2
Interfaz z
I_interfaz_1
al_emparejado_1
al_emparejado_2
Var
Var
Var
Var

FIG 14a

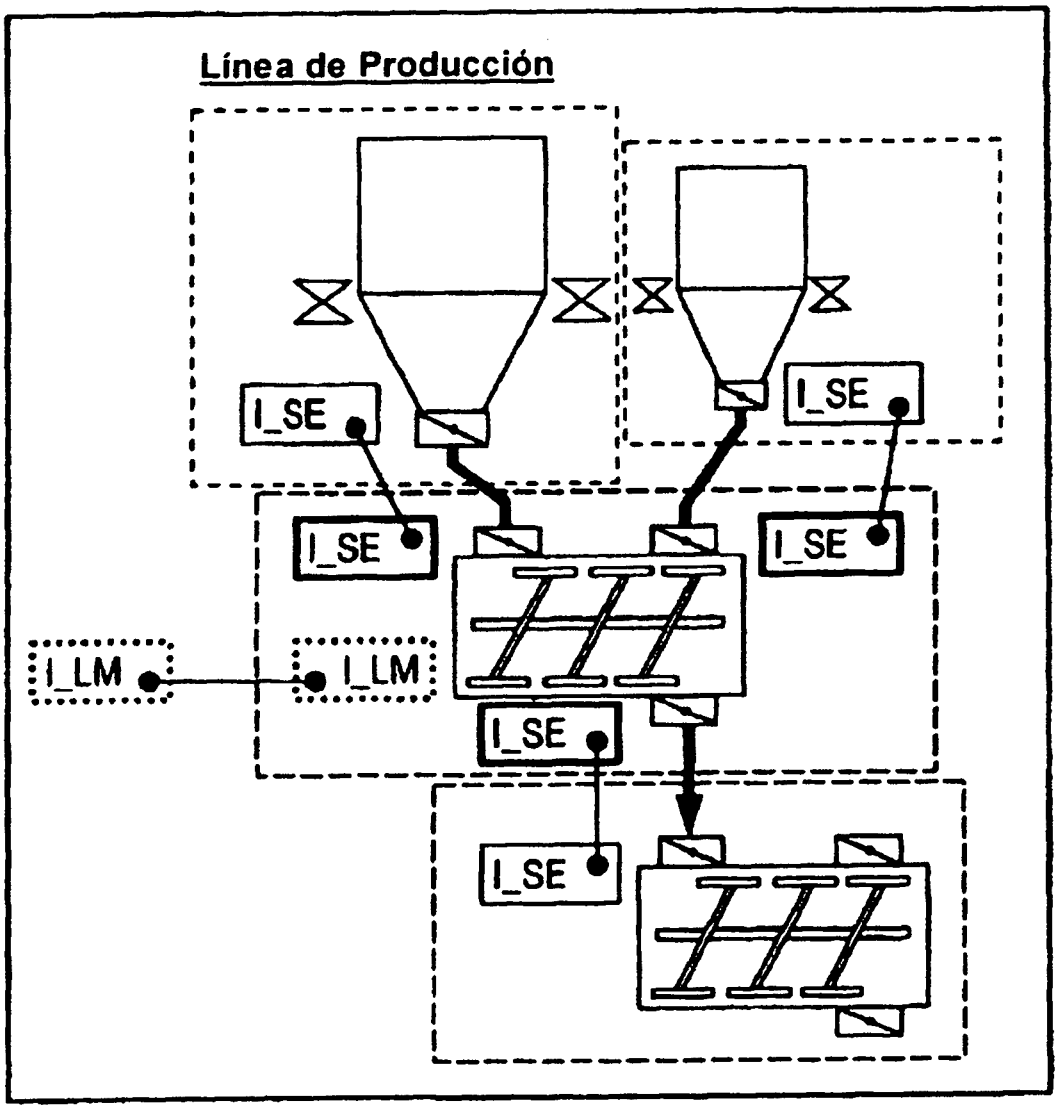
Línea de Producción
I_SE
I_SE
I_SE
I_SE
I_SE
I_LM
I_LM
I_SE
I_SE
I_SE = Interfaz emisor-receptor
I_LM = Interfaz línea-máquina

FIG 14b