(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 997 967**

(51) Int. CI.:

G06F 11/10 (2006.01)
G06F 11/07 (2006.01)
G06F 11/14 (2006.01)
G06F 11/16 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **22.04.2022 E 22169515 (8)**

(97) Fecha y número de publicación de la concesión europea: **04.09.2024 EP 4266175**

(54) Título: **Procedimiento para la operación apoyada por ordenador de una unidad de almacenamiento de datos y ejecución de programas de aplicación con verificación de errores de memoria de una memoria de almacenamiento**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
18.02.2025

(73) Titular/es:

SIEMENS MOBILITY GMBH (100.00%)
Otto-Hahn-Ring 6
81739 München, DE

(72) Inventor/es:

GERKEN, STEFAN;
SCHALLENBERG, ANDREAS y
SEEMANN, MARKUS

(74) Agente/Representante:

CARVAJAL Y URQUIJO, Isabel

DESCRIPCIÓN

Procedimiento para la operación apoyada por ordenador de una unidad de almacenamiento de datos y ejecución de programas de aplicación con verificación de errores de memoria de una memoria de almacenamiento

La invención se refiere a un método para el funcionamiento asistido por ordenador de una unidad de almacenamiento, en el que los datos se almacenan en la unidad de almacenamiento y antes del almacenamiento los datos se codifican, los datos se recuperan de la unidad de almacenamiento y después de la recuperación los datos se decodifican, la unidad de almacenamiento siendo monitoreada para detectar errores mediante la realización de una secuencia temporal de ejecuciones de prueba asistidas por ordenador para la unidad de almacenamiento. La invención también se refiere a un método para la ejecución asistida por ordenador de un programa de aplicación, en el que se opera una unidad de almacenamiento de la manera descrita anteriormente y en el que los conjuntos de datos de aplicación necesarios para la ejecución se recuperan de la unidad de almacenamiento y se decodifican. Finalmente, la invención se refiere a un producto de programa de ordenador y a un dispositivo de provisión para este producto de programa de ordenador, estando equipado el producto de programa de ordenador con instrucciones de programa para llevar a cabo este método.

En principio, los sistemas relevantes para la seguridad que funcionan con soporte informático en un entorno informático relevante para la seguridad deben garantizar que no se produzcan errores al recuperar datos de una unidad de almacenamiento. No se podrán utilizar datos corruptos para el procesamiento de datos. Relevante para la seguridad en el sentido de esta invención significa relevancia para la seguridad operativa del sistema (también denominada *Safety*). Por ejemplo, las aplicaciones ferroviarias se consideran sistemas relevantes para la seguridad en el sentido del término mencionado.

Para proteger entornos informáticos relevantes para la seguridad, se sabe que en varias instancias informáticas de un ordenador sólo se instalan de forma segura los componentes que sirven para procesar la aplicación en cuestión. Si luego la aplicación se ejecuta en paralelo en varias instancias informáticas, se puede utilizar una *voting* de manera conocida para determinar si se producen errores en el cálculo, lo que también puede revelar errores de memoria.

Según el documento WO 2014/00924382 se describen un método y un ordenador que presenta un algoritmo de prueba para procesar aplicaciones en el entorno informático formado por el ordenador. Los programas de aplicación se instalan en el ordenador y se ejecutan de forma redundante, de modo que el algoritmo de prueba tiene la oportunidad de detectar errores comparando los resultados del cálculo (como se mencionó anteriormente, también llamado *voting*). Sin embargo, este método requiere una cantidad significativa de *hardware* y, en determinadas situaciones, no permite una detección de errores fiable.

El documento DE 10 2007 040721 describe además una instalación de procesamiento de datos con una memoria que presenta varios elementos de memoria para almacenar cada uno una palabra de datos. A cada elemento de almacenamiento de la pluralidad de elementos de almacenamiento se le asigna información de firma que se basa en la información de dirección asignada al elemento de almacenamiento. Una palabra de código se asigna a una palabra de datos representando la palabra de datos y la información de firma como valores numéricos y la palabra de código se forma a partir de la palabra de datos y la información de firma mediante una operación aritmética.

El objetivo de la invención es proporcionar un método para el funcionamiento asistido por ordenador de una unidad de memoria y un método para la ejecución asistida por ordenador de un programa de aplicación en el que se utiliza dicha unidad de memoria, con el que se puede mejorar la fiabilidad de una memoria en cuanto a errores de memoria. Además, el objetivo de la invención es proporcionar un producto de programa informático y un dispositivo de provisión para este producto de programa informático con el que se pueda llevar a cabo el método antes mencionado.

Este objetivo se consigue según la invención con el objeto de la reivindicación indicado al principio (procedimiento para el funcionamiento asistido por ordenador de una unidad de almacenamiento), que codificando los datos por primera vez

- se genera o selecciona al menos un registro de datos de aplicación que contiene secciones de datos con datos de aplicación para un programa de aplicación y secciones de datos de prueba,
- para cada registro de datos de aplicación, a la sección de datos de prueba se le asignan datos de recuento (ZD), que identifican la ejecución de prueba que se está llevando a cabo.
- cada registro de datos de la aplicación está codificado y almacenado, y eso para verificar los datos en la ejecución de prueba que se lleva a cabo después de recuperar y decodificar los registros de datos de la aplicación

• se detecta un error para un conjunto de datos de la aplicación si los datos de recuento (ZD) no identifican ni la ejecución de prueba que se está llevando a cabo actualmente ni la ejecución de prueba completada más recientemente,
• la sección de datos de prueba de la sección de datos de la aplicación correspondiente se llena con datos de recuento (ZD), que identifican la ejecución de prueba que se está llevando a cabo, si no se detectó ningún error,
• el registro de datos de la aplicación correspondiente se codifica y se vuelve a almacenar si no se detecta ningún error.

Debido a que a cada una de las secciones de datos se le asigna una sección de datos de prueba (que se creó previamente si aún no había una sección de datos de prueba, o se seleccionó si una sección de datos de prueba ya estaba disponible para la sección de datos relevante) al codificar por primera vez, se pueden guardar los datos de conteo de registros relevantes, que contienen información sobre la ejecución de la prueba en curso. A través de la codificación posterior del conjunto de datos en cuestión se protegen los datos de recuento, así como los demás datos de la sección de datos en cuestión, desde el acceso hasta la descodificación. En el estado codificado, los datos no se pueden cambiar accidentalmente (por ejemplo, debido a un mal funcionamiento) o intencionalmente (por ejemplo, debido a un ataque externo) sin que un cambio se reconozca como un error cuando se decodifican los datos. Al menos es extremadamente improbable que el cambio no se note como un error cuando se decodifica el conjunto de datos. En este sentido, no existe protección contra cambios per se, sino sólo contra cambios no detectados. Al comprobar cíclicamente el contenido de la memoria para una codificación correcta (utilizando los datos de conteo), también se detectan los llamados errores durmientes. Se trata de errores que pueden producirse durante el almacenamiento de datos, por ejemplo, debido a cambios físicos en el medio de almacenamiento (inflexión de bits, etc.). Según la invención, la baja probabilidad de no descubrir errores permite mantener niveles de seguridad más elevados (SEL-1 ... SIL-4).

Todos los datos que requiere el programa de aplicación están preferentemente protegidos en el registro de datos, es decir, tanto los datos que componen el propio programa como los datos que representan la información a procesar.

Por ejemplo, es posible que, debido a errores de almacenamiento, es decir, errores que se producen durante el almacenamiento de los datos en una unidad de almacenamiento (por ejemplo, los llamados inflexores de bits) o debido a errores de procesamiento, es decir, errores que se producen durante el procesamiento de los datos. registro de datos, se modifican los datos de conteo. Sin embargo, dicho cambio se detecta cuando se realiza la siguiente ejecución de prueba si los datos del recuento no identifican ni la ejecución de prueba que se está llevando a cabo actualmente ni la ejecución de prueba completada más recientemente. En tal caso, se detecta un error. Aunque la causa tal vez no esté claramente determinada y pueda ser inofensiva para el tratamiento de los datos, por razones del nivel de seguridad que se desea alcanzar, una vez identificado el error, se emitirá y/o se tomarán medidas para impedir el posterior tratamiento de los datos incorrectos.

Sólo si no se detecta ningún error, se les asignan los datos de conteo de la ejecución de prueba en curso a los datos de conteo de la sección de datos de prueba, se codifica el registro de datos y se almacena nuevamente. Esto significa que el registro de datos en cuestión se marca simultáneamente como verificado en la ejecución de prueba correspondiente (más adelante se explicará la posibilidad de asignar los datos de recuento actuales a los registros de datos fuera de una ejecución de prueba en la sección de datos de prueba).

Los datos de conteo deben tener la propiedad de formar una serie en la que se conozca el predecesor y el sucesor de cada elemento del conjunto de datos de conteo. En principio, se pueden utilizar todas las series matemáticas. De manera especialmente preferente se puede seleccionar el conjunto de números naturales.

Para realizar la ejecución de prueba se utiliza preferentemente un programa de aplicación (programa de utilidad), que realiza la ejecución de prueba. Para ello, el programa de aplicación en cuestión, que también puede denominarse programa de prueba, accede a la unidad de almacenamiento, descodifica, comprueba y codifica registro tras registro de datos hasta finalizar una ejecución de prueba. Si se detecta un error, el programa de prueba puede generarlo. El programa de prueba también puede contener funciones que contienen una reacción a un error identificado, por ejemplo, la suspensión de un programa de aplicación (programa de utilidad) que utiliza las secciones de datos incorrectas de la aplicación (llamadas secciones de datos de la aplicación) y, por lo tanto, podría potencialmente hacer errores relevantes para la seguridad. Esto hace visibles diversas fuentes de error.

La ejecución de la corrida de prueba se puede controlar, por ejemplo, accediendo a los registros de datos individuales. Si las direcciones de todos los registros de datos se conocen al comienzo de la ejecución de prueba, los registros de datos se pueden llamar secuencialmente hasta que se complete la ejecución de prueba (más sobre esto a continuación).

Si una ejecución de verificación en su conjunto se realiza incorrectamente, en el sentido de que no se verifican todos los registros de datos en una ejecución de verificación, esto se notará a más tardar en la siguiente ejecución de verificación, al menos siempre que se verifiquen los registros de datos no verificados nuevamente

en la siguiente ejecución de verificación. Estos todavía contienen los datos de conteo de la penúltima ejecución de prueba, lo que se nota en la siguiente ejecución de prueba. Esto hace que se detecte y emita un error. Por lo tanto, los datos de recuento también permiten controlar la correcta ejecución de las propias ejecuciones de prueba. Esto permite identificar otra fuente de error. Los datos clasificados como obsoletos y, por tanto, potencialmente incorrectos (porque no se controlan periódicamente en una ejecución de prueba) no están suficientemente seguros en términos de su integridad, por lo que se consideran un error.

Esto crea un mecanismo de seguridad que aumenta el funcionamiento de la unidad de almacenamiento y la ejecución de los programas de aplicación (estos son programas de servicio o programas de utilidad, más sobre esto a continuación) en el sentido de la tarea. De este aumento se benefician especialmente las aplicaciones relevantes para la seguridad, por ejemplo, las aplicaciones ferroviarias, que deben cumplir determinados niveles de seguridad (también llamados niveles de seguridad, SIL-1 ... SIL-4 o *Safety Integrity Level*) como requisito de aprobación. En el contexto de esta invención, el término "seguridad" debe entenderse en el sentido de seguridad operativa (*Safety*). En particular, la codificación se realiza principalmente desde el punto de vista de la seguridad operativa y no desde el punto de vista de la seguridad de la transmisión (*Security*). Por lo tanto, según la invención se prevé preferentemente un método de codificación que consiga un alto rendimiento durante la codificación y decodificación (con el efecto de tiempos cortos de codificación y decodificación) y que no garantice un alto nivel de dificultad al descifrar una codificación sin autorización.

En general, con el procedimiento según la invención para el funcionamiento de la unidad de almacenamiento y para la ejecución de programas de aplicación se puede determinar qué conjuntos de datos están realmente o al menos potencialmente corruptos y, por lo tanto, ponen en peligro la seguridad operativa durante la ejecución de los programas de aplicación, especialmente los programas de utilidad. Los conjuntos de datos afectados se marcan con un error y preferiblemente se excluyen del procesamiento de datos posteriores.

En otras palabras, según la invención la fiabilidad operativa está, por así decirlo, ligada a los propios conjuntos de datos. Para ello, según la invención, estos están provistos de secciones de datos de prueba adecuadas. Esto hace posible ventajosamente que los conjuntos de datos se almacenen en una misma unidad de almacenamiento, incluso si los procesos se ejecutan en paralelo (de forma redundante) por razones de seguridad operativa. En particular, se pueden utilizar componentes de *software* y *hardware* disponibles comercialmente, los llamados componentes COTS (COTS significa *Commercial off-the-shelf*), para almacenar los conjuntos de datos y procesar los programas de aplicación sin poner en peligro los requisitos de seguridad operativos. Por lo tanto, estos componentes también se pueden utilizar, por ejemplo, en aplicaciones de tecnología ferroviaria y pueden sustituir sistemas propietarios. La ventaja de los componentes COTS es que su compra es económica y también pueden reemplazarse más fácilmente si es necesario e integrarse en soluciones en la *cloud*.

En el contexto de la invención, por "asistida por ordenador" o "implementada por ordenador" se puede entender una implementación del método en la que al menos un ordenador o procesador lleva a cabo al menos un paso del método.

En el contexto de la invención, un "entorno informático" puede entenderse como una infraestructura que consta de componentes tales como ordenadores, unidades de almacenamiento, programas y datos a procesar con los programas, que se utilizan para ejecutar al menos una aplicación que debe cumplir una tarea. En particular, la infraestructura también puede consistir en una red de los componentes mencionados.

Una “instancia informática” (o instancia para abreviar) puede entenderse como una unidad funcional dentro de un entorno informático que puede asignarse a una aplicación y puede ejecutarla. Cuando se ejecuta la aplicación, esta unidad funcional forma un sistema físicamente y/o virtualmente autónomo.

El término "ordenador" u "computadora" cubre cualquier dispositivo electrónico con propiedades de procesamiento de datos. Por ordenadores pueden ser, por ejemplo, ordenadores personales, servidores, ordenadores portátiles, dispositivos móviles y otros dispositivos de comunicación que procesan datos con soporte informático, procesadores y otros dispositivos electrónicos para el procesamiento de datos, que preferentemente también pueden estar conectados en red a través de interfaces.

En el marco de la invención se entiende por procesador, por ejemplo, un convertidor, un sensor para generar señales de medición o un circuito electrónico. Un procesador puede ser en particular un procesador principal (en inglés *Central Processing Unit*, CPU), un microprocesador, un microcontrolador o un procesador de señales digitales, posiblemente en combinación con una unidad de memoria para almacenar instrucciones y datos de programas. Un procesador también puede entenderse como un procesador virtualizado o una *Soft*-CPU.

Por “unidad de memoria” se entiende en el marco de la invención, por ejemplo, una memoria legible por ordenador en forma de memoria de acceso aleatorio (en inglés *Random-Access Memory,* RAM) o memoria de datos (disco duro o soporte de datos).

Las "interfaces" se pueden implementar en términos de *hardware*, por ejemplo, cableado o como conexión inalámbrica, y/o *software*, por ejemplo, como interacción entre módulos de programa individuales o partes de programa de uno o varios programas de ordenador.

Una "*cloud*" debe entenderse como un entorno para la "C*loud-Computing*" (también denominada nube informática o nube de datos). Se refiere a una infraestructura de IT que está disponible a través de interfaces de una red como Internet. Suele incluir espacio de almacenamiento, potencia informática o software como servicio sin necesidad de instalarlo en el ordenador local mediante la *cloud*. Los servicios ofrecidos como parte de la computación en la *cloud* cubren todo el espectro de las tecnologías de la información e incluyen, entre otras cosas, infraestructura, plataformas y *software*.

Por "módulos de programa" deben entenderse unidades funcionales individuales que permiten una secuencia de programas de pasos del método según la invención. Estas unidades funcionales pueden implementarse en un único programa informático o en varios programas informáticos que se comunican entre sí. Las interfaces implementadas aquí se pueden implementar en términos de *software* dentro de un solo procesador o en términos de *hardware* si se utilizan varios procesadores.

A menos que se especifique lo contrario en la descripción siguiente, los términos "crear", "determinar", "calcular", "generar", "configurar", "modificar" y similares se refieren preferentemente a procesos que generan datos y/o cambian y/o convierten los datos en otros datos. Los datos están disponibles en particular como cantidades físicas, por ejemplo, como impulsos eléctricos o cantidades eléctricas analógicas. Las instrucciones requeridas se resumen en un programa informático como *software*. Además, los términos "enviar", "recibir", "incorporar", "leer", "transmitir" y similares se refieren a la interacción de componentes de *hardware* y/o componentes de *software* individuales a través de interfaces.

Según una configuración de la invención está previsto que antes de realizar por primera vez una ejecución de prueba se ajuste un valor inicial para los datos de conteo.

El valor inicial debe ser un elemento de la serie numérica utilizada. Un número natural, especialmente cuando se utilizan números naturales como datos de conteo. Es ventajoso utilizar un número distinto de uno y, en particular cuando se utilizan varias unidades de almacenamiento, utilizar valores iniciales diferentes, porque esto reduce aún más la probabilidad de coincidencias aleatorias entre datos de conteo incorrectos de diferentes unidades de almacenamiento. Esto aumenta aún más ventajosamente la seguridad del proceso, ya que se reduce aún más la probabilidad de que se produzcan errores de memoria o de procesamiento no detectados.

Según una realización de la invención está previsto que cada ejecución de prueba incluya al menos la verificación de todos los conjuntos de datos de aplicación disponibles para el método actual al comienzo de la ejecución de prueba.

En esta configuración del método se determina al inicio de la prueba qué registros de datos deben comprobarse. A continuación, se comprueba el número determinado de registros de datos a comprobar. Si se han verificado todos los registros de datos a verificar, la ejecución de verificación habrá finalizado. Los datos de conteo se pueden asignar entonces al siguiente elemento de conteo. Al definir, por así decirlo, el alcance de la prueba de una ejecución de prueba y luego llevarla a cabo, el proceso de prueba se puede llevar a cabo ventajosamente de manera confiable y con poco esfuerzo computacional.

Alternativamente, también sería posible comprobar en una ejecución de comprobación únicamente aquellos registros de datos que se hayan modificado durante la ejecución de comprobación anterior. Sin embargo, esto implicaría monitorear los registros para detectar uso o cambios, lo que complicaría el proceso. Además, también pueden ocurrir errores de memoria, como cambios de bits, mientras no se accede a los datos almacenados. El cambio de registros de datos se analizará con más detalle a continuación.

En principio, es posible realizar una ejecución de verificación si todos los registros de datos a verificar no son requeridos (accedidos y/o modificados) por los programas de aplicación. Sin embargo, también es posible realizar la ejecución de prueba en paralelo con los programas de aplicación en ejecución, es decir, mientras los programas de aplicación acceden a todos los registros de datos bajo prueba. En este caso, los registros que se están editando actualmente no se pueden verificar (más sobre esto a continuación).

Según una configuración de la invención está previsto que, una vez finalizada una ejecución de prueba, a los datos de conteo se les asigne un elemento de conteo que sigue al elemento de conteo para la asignación anterior de los datos de conteo.

Se completó una ejecución de prueba cuando se verificaron todos los conjuntos de datos de la aplicación que contiene la ejecución de prueba. Esto significa que durante la ejecución de la prueba se supervisa qué conjuntos de datos de la aplicación ya se han comprobado y cuáles no. Esto se puede conseguir, por ejemplo, mediante las direcciones de los juegos de datos de aplicación, con las que se pueden identificar los juegos de datos de aplicación.

Si ha finalizado una ejecución de prueba y los datos de conteo puestos a disposición del programa de aplicación para la prueba se han asignado al elemento de conteo posterior, entonces el programa de aplicación puede recuperar los datos de conteo actualizados en la ejecución de prueba posterior. Al comprobar los registros de datos almacenados se puede comprobar entonces si los registros de datos contienen los datos de conteo de la ejecución de prueba anterior, ya que el programa de aplicación "conoce" el elemento de conteo de la ejecución de prueba actual, por así decirlo, aumentando previamente los datos del conteo. Dado que los datos de conteo se ajustan al elemento de conteo actual para cada registro de datos verificado (al menos si no se detectaron errores), las ejecuciones de prueba se pueden repetir tantas veces como se desee usando este procedimiento.

El objetivo planteado se consigue alternativamente con el objeto de la reivindicación indicado al principio (método para la ejecución asistida por ordenador de un programa de aplicación) según la invención, mediante el cual para la ejecución asistida por ordenador de un programa de aplicación, en el que una unidad de almacenamiento se opera como se explicó anteriormente y en el que los conjuntos de datos de la aplicación requeridos para la ejecución se recuperan y decodifican de la unidad de almacenamiento,

- después de decodificar los registros de datos de la aplicación, se ejecuta el programa de aplicación,
- al menos registros de datos de aplicación que se modifican o crean mediante la ejecución del programa de aplicación

Contienen secciones de datos, codificadas y almacenadas en la unidad de almacenamiento.

Preferiblemente, el programa de aplicación se puede ejecutar paso a paso. Esto significa que los conjuntos de datos de aplicación necesarios para ejecutar el programa de aplicación también deben recuperarse de la unidad de almacenamiento y codificarse al menos sólo para el siguiente paso del programa de aplicación. Por supuesto, a medida que se procesa el programa de aplicación, se recuperan y decodifican gradualmente la totalidad de los registros de datos de aplicación necesarios para procesar el programa de aplicación.

Debido a que la unidad de almacenamiento se opera como se describió anteriormente de acuerdo con el método para la operación asistida por ordenador de la unidad de almacenamiento, en otras palabras, los conjuntos de datos de aplicación que contienen secciones de datos modificadas o generadas mediante la ejecución del programa de aplicación (programa de utilidad) también se incluyen en la codificación de los datos de conteo que caracterizan la ejecución de prueba actual que se está llevando a cabo.

Esto tiene varias ventajas. Si se crean nuevos conjuntos de datos o se modifican (es decir, se modifican datos de un conjunto de datos existente, en particular las secciones de datos con datos de aplicación), estos se almacenan con datos de conteo adecuados para comprobar los conjuntos de datos correspondientes al menos en la siguiente prueba. Si los registros de datos almacenados no contienen errores, en la siguiente ejecución de prueba también se les proporcionarán los datos de recuento correspondientes para que puedan reconocerse como libres de errores.

Esta realización de la invención también tiene la ventaja de que los conjuntos de datos que no están disponibles para la prueba en una ejecución de prueba que se ejecuta paralelamente al procesamiento de datos mediante programas de aplicación pueden omitirse en la ejecución de prueba, ya que estos conjuntos de datos son procesados por un programa de aplicación en particular, mediante un programa de utilidad, completarse con datos de pago actuales que permitan una verificación en la ejecución de prueba posterior, incluso si se omite una verificación (por ejemplo, mediante un programa de prueba) en la ejecución de prueba actual debido al procesamiento paralelo por parte del programa de aplicación (programa de servicio o programa de utilidad). En particular, esto simplifica considerablemente el proceso de ejecución de la verificación, ya que en la ejecución de la verificación se pueden omitir simplemente registros de datos que se procesan durante la ejecución de la rutina de verificación y no es necesario realizar una verificación en un momento posterior a la ejecución de verificación actual. Desde el punto de vista de la seguridad, esto se puede representar porque la integridad de los datos, teniendo en cuenta los datos de recuento, también puede ser realizada por el programa de aplicación (programa de utilización) (o en un futuro previsible en la próxima ejecución de prueba) y así el registro de datos en cuestión se mantiene actualizado.

Según una forma de realización de la invención está previsto que al comprobar los datos después de decodificar los registros de datos de la aplicación y antes de ejecutar el programa de aplicación

- se detecta un error para un registro de datos de aplicación si los datos de recuento no identifican ni la ejecución de prueba que se está llevando a cabo actualmente ni la ejecución de prueba completada más recientemente,
- el programa de aplicación se ejecuta si no se detecta ningún error.

En esta realización de la invención, las ventajas que se logran al verificar periódicamente los datos en la unidad de almacenamiento mediante ejecuciones de prueba también se logran cuando los datos son procesados por un programa de aplicación (programa de usuario). En otras palabras, cada vez que se recuperan datos de la unidad de almacenamiento para su procesamiento, se comprueba si mientras tanto se ha producido un error de almacenamiento. De este modo se consigue una seguridad adicional, ya que también se pueden detectar

errores de memoria que se hayan producido aleatoriamente en conjuntos de datos entre la última ejecución de prueba exitosa y la llamada del conjunto de datos por parte de un programa de aplicación (en particular, un programa de usuario).

Según una configuración de la invención está previsto que, durante el funcionamiento asistido por ordenador de la unidad de almacenamiento para la primera codificación de los datos, adicionalmente

• al menos un grupo de conjuntos de datos de aplicación, que contiene secciones de datos con datos de aplicación idénticos para un programa de aplicación y secciones de datos de prueba, cada una con diferentes características de diversidad, se genera o selecciona de una reserva de posibles características de diversidad,
• cada registro de datos de aplicación está codificado y almacenado,

y la recuperación de los datos se realiza recuperando y decodificando los registros de la aplicación, así como los datos se almacenan codificando y almacenando los conjuntos de datos de la aplicación,
y/o
que, durante la ejecución asistida por ordenador, el programa de aplicación se ejecuta varias veces en instancias informáticas redundantes, recuperándose y decodificando cada registro de datos de aplicación de un grupo codificado de registros de datos de aplicación de la unidad de almacenamiento para ejecutar los conjuntos de datos de aplicación requeridos, por lo cual
• para verificar los datos después de decodificar los conjuntos de datos de la aplicación, se detecta un error para un conjunto de datos de la aplicación si las características de diversidad en el conjunto de datos de la aplicación difieren de todas las características de diversidad posibles de esta instancia informática de la reserva, es decir, de la reserva destinado a esta instancia informática,
• el programa de aplicación con las secciones de datos de cada registro de datos de aplicación del grupo correspondiente se lleva a cabo si no se ha detectado ningún error,
y/o
• al menos los conjuntos de datos de aplicación, que contienen secciones de datos modificadas o generadas mediante la ejecución del programa de aplicación, se codifican de la manera mencionada anteriormente y se almacenan en la unidad de almacenamiento.

Como se describirá a continuación, esta realización de la invención es un mecanismo de seguridad adicional que aumenta aún más el funcionamiento de la unidad de almacenamiento y la ejecución de programas de aplicación en el sentido de la tarea. Como ya se ha explicado, de este aumento se benefician especialmente las aplicaciones relevantes de seguridad, como por ejemplo las aplicaciones ferroviarias. Sólo tiene que ocurrir uno de los errores descritos en el contexto de esta invención para provocar que se genere un error general. Independientemente de cuál de las medidas según la invención se utilice para detectar el error, aumenta la seguridad al ejecutar los programas de aplicación (programas de utilidad). Como ya se ha mencionado, el término seguridad debe entenderse en el sentido de seguridad operativa.

La formación de grupos de conjuntos de datos de aplicación, cada uno de los cuales contiene secciones de datos con datos de aplicación idénticos, tiene la ventaja de que se pueden realizar de forma redundante operaciones aritméticas, es decir, el procesamiento de programas de aplicación que utilizan los mismos datos de aplicación. Si se identifica un error en uno de los conjuntos de datos de la aplicación, el conjunto de datos de la aplicación relevante se puede excluir del procesamiento sin que el proceso tenga que interrumpirse inmediatamente. Porque para la realización del procedimiento, es decir, para la elaboración del programa de solicitud en cuestión, se dispone de conjuntos de datos de solicitud adicionales.

Las características de diversidad hacen que los conjuntos de datos de aplicación, cada uno con datos de aplicación idénticos, se puedan distinguir entre sí y permiten una asignación precisa, por ejemplo, a una instancia informática destinada a los datos de aplicación en cuestión. Esto también permite comprobar si se ha producido un error (lo que se denomina error de asignación) al asignar conjuntos de datos de la aplicación a determinadas instancias informáticas. Un error identificado de este tipo también puede provocar reacciones adecuadas en el proceso. Por ejemplo, una asignación incorrecta de conjuntos de datos de aplicación puede provocar que las instancias informáticas correspondientes queden excluidas de la realización del procedimiento. También en este caso las instancias informáticas restantes pueden continuar ejecutando el proceso, por lo que el proceso no se interrumpe inmediatamente.

Si para un programa de aplicación se necesitan varios conjuntos de datos de aplicación (y ésta es la regla), cada grupo de conjuntos de datos de aplicación puede recibir en particular las mismas características de diversidad. Estos se seleccionan de una reserva de características de diversidad, de esta manera las características de diversidad permiten que los conjuntos de datos de la aplicación se asignen a una instancia informática específica con una diversidad específica (en otras palabras, una instancia informática a la que se le asignan características de diversidad de un tipo específico del conjunto de características de la reserva). Esto aumenta significativamente la confiabilidad operativa y puede implementarse con poco esfuerzo computacional debido a la reserva limitada de características de diversidad.

Si los datos se mencionan generalmente en el contexto de esta solicitud, pueden tener cualquier contenido. Por el contrario, por datos de aplicación, características de diversidad, datos de redundancia, datos de conteo, etc. se entienden datos específicos, es decir, datos para un propósito específico. Además, se hace una distinción entre

- datos variables: estos datos se asignan o cambian según el tiempo de ejecución (especialmente los datos de la aplicación)
- datos estáticos: datos de programa (en particular datos de aplicación que describen programas de aplicación).

La sección de datos que contiene datos de aplicación (llamada abreviadamente sección de datos de aplicación) puede ejecutarse (datos estáticos como instrucciones para la ejecución del programa de aplicación) o procesarse (datos variables a procesar para el programa de aplicación). Una sección de datos de prueba, que contiene preferiblemente las características de diversidad y/o datos de redundancia y/o datos de recuento, se usa preferiblemente para el procesamiento mediante programas de utilidad que se ejecutan para verificar la integridad de los datos (más sobre esto a continuación).

El almacenamiento codificado de los registros de datos de aplicación aumenta ventajosamente la probabilidad de identificar errores de datos durante el procesamiento (errores de procesamiento) o durante el almacenamiento (errores de memoria). Incluso un simple cambio de bit, por ejemplo, durante el almacenamiento de datos, conduciría a un resultado diferente durante la decodificación, de modo que el conjunto de datos de aplicación codificados difiere significativamente del conjunto de datos de aplicación a codificar. Si se descubre un error de este tipo, se llevan a cabo las medidas ya mencionadas para que el proceso garantice la seguridad operativa requerida.

El uso de las características de diversidad para identificar errores al procesar los conjuntos de datos de la aplicación mediante un programa de aplicación ya se ha explicado con más detalle anteriormente. Para procesar los registros de datos de la aplicación, estos se decodifican y codifican nuevamente antes de volver a almacenarlos en la unidad de almacenamiento. También es posible cambiar los conjuntos de datos de la aplicación ejecutando el programa de aplicación o formar conjuntos de datos de la aplicación con secciones de datos recién creadas. Luego, estos se codifican utilizando el método de almacenamiento mencionado anteriormente y se almacenan en la unidad de almacenamiento.

Al comprobar las características de diversidad se comprueba al menos si las características de diversidad reconocidas de un conjunto de datos de aplicación difieren de todas las características de diversidad posibles (es decir, posibles para la instancia informática correspondiente) de la reserva. Entonces se detecta un error; si éstas corresponden a las características de diversidad esperadas, se supone que no hay error. Preferiblemente, sólo se permite un tipo de datos de características de diversidad de la reserva para una instancia informática. Sin embargo, también es posible reservar las instancias informáticas en diferentes áreas para utilizar mejor las instancias informáticas para diferentes características de diversidad (más sobre esto a continuación).

Al comprobar los datos se puede comprobar, por ejemplo, si las características de diversidad en el conjunto de datos de la aplicación difieren de todas las características de diversidad posibles de la reserva, si es fundamentalmente posible que todas las características de diversidad de la reserva se puedan utilizar para una determinada instancia informática. Sin embargo, también es posible que determinadas características de diversidad estén reservadas de la reserva para una instancia informática específica. Entonces sólo se comprueba si las características de diversidad difieren de estas características de diversidad reservadas de la instancia informática. Esto aumenta ventajosamente la probabilidad de detectar un error al detectar características de diversidad incorrectas.

Según una realización de la invención, está previsto que las diferentes características de diversidad de la reserva contengan diferentes constantes de codificación y constantes de decodificación.

Las constantes de codificación se usan en un algoritmo de codificación y las constantes de decodificación se usan en un algoritmo de decodificación. Dichos algoritmos de codificación y algoritmos de decodificación son conocidos en sí. Usando las constantes, se puede implementar ventajosamente la codificación y decodificación con alto rendimiento. Se trata sólo de lograr la seguridad operativa (*safety*), no de lograr una seguridad suficiente contra ataques enemigos (*security*), lo que sólo sería posible con un esfuerzo informático significativamente mayor y la consiguiente pérdida de rendimiento informático.

Si las constantes de codificación y las constantes de decodificación se utilizan simultáneamente como características de diversidad, esto tiene la ventaja de que la diversidad de los conjuntos de datos de aplicación se incluye en la codificación. Esto significa que para codificar y decodificar se utilizan un par de constantes que definen al mismo tiempo la diversidad del conjunto de datos de aplicación en cuestión. Con las constantes se cumplen dos funcionalidades al mismo tiempo, lo que ahorra esfuerzo de cálculo y por tanto aumenta el rendimiento del método.

Según una realización de la invención, se prevé que las características de diversidad incluyan un operador de adaptación para adaptar la sección de datos generada mediante decodificación a la sección de datos antes de la decodificación.

El operador de adaptación es necesario si el contenido de información (por ejemplo, ancho de bits) del conjunto de datos de aplicación codificados y posteriormente descodificados es mayor que el del conjunto de datos de aplicación codificados. Las informaciones superfluas no se pueden tener en cuenta después de que el juego de datos de aplicación haya sido descodificado por la aplicación que se va a ejecutar y, por así decirlo, el operador de adaptación debe cortarlas. En general, esto da como resultado la ecuación funcional para un algoritmo de codificación y decodificación

$$((AD\ opA\ C\_enc)\ opB\ C\_dec)\ opC\ Y == AD$$

con

AD	conjunto de datos de la aplicación
opA	operador de codificación
C_enc	constante de codificación
opB	operador de decodificación
C_des	constante de decodificación
opC	operador de ajuste
Y	contenido de información deseado (ancho de bits)

Según una forma de realización de la invención está previsto que el operador de codificación sea una instrucción con la que se realiza una multiplicación.

Según otra configuración de la invención está previsto que el operador de decodificación (y en particular también el operador de codificación) sea una instrucción con la que se realiza una multiplicación como operación de decodificación (y en particular como operación de codificación) y el operador de adaptación es adecuado para esto después de decodificar la sección de datos para limitar su longitud original (ancho de bits).

Si se selecciona una multiplicación como operador de decodificación (y en particular también como operador de codificación), se trata de una operación que puede implementarse ventajosamente en un ordenador de alto rendimiento. En particular, si la constante de descodificación (y en particular también la constante de codificación) es mayor que 1 y es un número entero, el proceso se puede realizar de forma especialmente sencilla desde el punto de vista computacional. Sin embargo, la doble multiplicación aumenta la longitud de la sección de datos codificados y luego decodificados. Después de la decodificación, el contenido de información real de la sección de datos debe, por así decirlo, separarse del resultado completo de la decodificación.

Para este fin es adecuada preferentemente una operación de módulo mod (también denominada operación de restricción de longitud), que corta ventajosamente los bits sobrantes en expresiones binarias hasta un ancho de bits Y deseado sin cambiar el contenido de los bits del residuo restante. Para almacenar la información sobrante (información redundante) en estado codificado (ya que la decodificación sólo es posible con ayuda de esta información redundante), se debe proporcionar espacio de almacenamiento adicional para estos datos (bits) en el conjunto de datos de la aplicación. Para aumentar aún más la seguridad del proceso, los datos del espacio de almacenamiento adicional también se pueden comprobar en busca de errores como datos de redundancia (más información a continuación).

Teniendo en cuenta la multiplicación como operación de codificación y decodificación y la operación de módulo como operador de ajuste, resulta la siguiente expresión para la ecuación funcional anterior para codificación y decodificación:

$$(AD \cdot C\_enc \cdot C\_dec)\ mod\ Y == AD$$

con

AD	conjunto de datos de la aplicación
C_enc	constante de codificación > 1
C_dec	constante de decodificación > 1
Mod Y	operación de módulo
Y	ancho de bit deseado

La constante de codificación y la constante de decodificación son un par de números que caracterizan especialmente también la diversidad del conjunto de datos de aplicación en cuestión. Este par de números se selecciona de modo que cuando el conjunto de datos de aplicación se multiplica por la constante de codificación y el conjunto de datos de aplicación codificados se multiplica entonces por la constante de decodificación, los

bits del ancho de bits Y deseado de los datos de aplicación vuelven a tener los mismos valores por bit. En la reserva de datos de características de diversidad, para cada diversidad están disponibles una constante de codificación y una constante de decodificación, cada una de las cuales forma un par de números con las propiedades mencionadas. Los experimentos han demostrado que un algoritmo de codificación y decodificación que utiliza la última ecuación funcional se puede llevar a cabo con un rendimiento comparativamente alto en términos de velocidad de cálculo.

Según una configuración de la invención está previsto que antes de la codificación la sección de datos de prueba se cree de tal manera que presente adicionalmente espacio de almacenamiento redundante con datos de redundancia predeterminados.

Como se describirá a continuación, esta realización de la invención es un mecanismo de seguridad adicional que aumenta aún más el funcionamiento de la unidad de almacenamiento y la ejecución de programas de aplicación en el sentido de la tarea. Como ya se ha explicado, este aumento se beneficia especialmente de las aplicaciones relevantes para la seguridad.

La ganancia en seguridad reside en el hecho de que los datos de redundancia especificados en el proceso también se conocen porque están especificados. Cuando se trata de datos redundantes, redundante significa que no es necesario almacenar los datos de la aplicación. Sin embargo, los datos de redundancia son necesarios para la codificación porque aumenta el volumen de datos (ancho de bits). Los datos de redundancia están ocupados por datos nuevos que normalmente difieren de los datos de redundancia especificados. Estos nuevos datos son necesarios para la decodificación posterior.

Sin embargo, si se utiliza un algoritmo de codificación o decodificación correspondiente, los datos de redundancia se completarán nuevamente con los datos especificados originalmente después de la decodificación. Esto significa que se detectará un error si no se cumple esta condición.

Preferiblemente, los datos especificados pueden contener sólo ceros o sólo unos. De este modo, el llenado de datos determinados se puede realizar de forma especialmente sencilla y con poco esfuerzo informático.

Según una forma de realización de la invención está previsto que las secciones de datos de aplicación estén compuestas por palabras de datos.

Una palabra de datos es una cantidad específica de datos que una computadora puede procesar en un solo paso en la unidad aritmeticológica del procesador. Si se trata de una cantidad máxima de datos, su tamaño se denomina ancho de palabra, ancho de procesamiento o ancho de bus. Una palabra de datos puede tener preferentemente un ancho de palabra de 32, 64 o 128 bits. Si las secciones de datos de aplicación se componen de palabras de datos, esto acelera ventajosamente el procesamiento de los distintos pasos del método, por lo que el método según la invención puede llevarse a cabo con un rendimiento óptimo.

Según una configuración de la invención está previsto que el procedimiento se lleve a cabo con al menos dos procesadores o núcleos de procesador al mismo tiempo.

Como ya se mencionó, los procesadores pueden ser físicos (incluso con núcleos de procesador) o virtuales (por ejemplo, emulados). Ventajosamente se pueden utilizar varios procesadores o núcleos de procesador, en particular para realizar procesos informáticos de forma redundante, es decir en paralelo y en particular simultáneamente, para conseguir una seguridad adicional frente a la aparición de errores en el procesamiento de los conjuntos de datos de la aplicación. La aparición de errores se puede comprobar, por ejemplo, mediante *voting*, como se explica más detalladamente a continuación.

Según una forma de realización de la invención está previsto que al menos dos procesadores o núcleos de procesador accedan conjuntamente a una misma unidad de memoria.

Acceder a una misma unidad de memoria significa que al menos dos procesadores/núcleos de procesador utilizan una unidad de memoria común, por lo que la unidad de memoria no proporciona ninguna separación física de áreas de memoria para un procesador u otro procesador. En otras palabras, cada procesador básicamente puede acceder a toda el área de memoria puesta a disposición por la unidad de memoria. Por supuesto, esto no excluye que determinadas áreas de memoria se reserven para uno u otro procesador, por ejemplo, definiendo direcciones de memoria. Sin embargo, esta reserva se puede cambiar o cancelar sin intervención del *hardware*, lo que potencialmente hace que toda el área de memoria esté disponible para cada uno de los procesadores.

El uso de unidades de memoria compartida para múltiples procesadores facilita ventajosamente el uso de componentes COTS. Además, comprar una unidad de memoria compartida es más rentable que proporcionar una unidad de memoria independiente para cada procesador. Esto permite implementar soluciones especialmente rentables.

Según una realización de la invención, se proporciona que

· la *voting* se realiza con secciones de datos idénticas guardadas varias veces y/o con secciones de datos que han sido modificadas de manera idéntica varias veces,
· Si se identifica un error como resultado de la *voting*, los conjuntos de datos de la aplicación que contienen secciones de datos que son responsables de la identificación del error se excluyen del procesamiento posterior.

Cuando los conjuntos de datos de la aplicación se almacenan y/o procesan de forma redundante, se producen secciones de datos idénticas guardadas varias veces y/o secciones de datos modificadas de forma idéntica varias veces. Los datos redundantes (es decir, las secciones de datos de la aplicación relevantes) deben ser idénticos, por lo que cualquier diferencia que se produzca se puede identificar y generar como error mediante *voting*.

La realización de *voting* para identificar errores en los datos es de por sí conocida. Por ejemplo, es posible calcular conjuntos de datos de aplicación independientes en varias secuencias de proceso paralelas, preferiblemente tres, y luego compararlas en el marco de una *voting*. Los resultados sólo se consideran fiables si al menos la mayoría de los resultados comparados coinciden. El procesamiento paralelo de una aplicación por parte del entorno informático puede realizarse en particular mediante tres conjuntos de datos de aplicación procesados en paralelo en el entorno informático. Esto puede garantizar ventajosamente que el procesamiento de un conjunto de datos pueda llevarse a cabo sin realimentación en relación con las otras operaciones de procesamiento. Esto puede reducir ventajosamente tanto como sea posible la probabilidad de que la aparición de errores en el procesamiento de un conjunto de datos de la aplicación afecte también al procesamiento de otros conjuntos de datos.

Si se detecta un error en los datos durante una ejecución de prueba o al ejecutar el programa de aplicación o un error de procesamiento durante la *voting*, se puede impedir o al menos suspender el procesamiento posterior de los datos o sólo impedir o al menos suspender en el caso de solicitudes que alcancen o superen un límite predeterminado. nivel de seguridad. Esto tiene la ventaja de que la capacidad informática de los ordenadores *host* afectados sigue estando disponible para aplicaciones que no son relevantes para la seguridad. Esto significa que las aplicaciones pueden llevarse a cabo si los errores que se producen (por ejemplo, una caída del programa) no suponen ningún riesgo relevante para la seguridad (los riesgos relevantes para la seguridad en aplicaciones ferroviarias son, por ejemplo, colisiones de trenes u otros accidentes).

Ventajosamente puede estar previsto que el primero y el segundo de los programas de aplicación se detengan si las secciones de datos a comparar no coinciden o los datos de recuento en las secciones de datos de prueba que pertenecen a estas secciones de datos no identifican ni la ejecución de prueba que se está realizando. realizado o la ejecución de prueba completada más recientemente.

Detener la secuencia del programa impide de forma ventajosa y eficaz que continúe el procesamiento incorrecto de la solicitud. El programa de aplicación en cuestión (incluidos los programas parciales y los complejos de programas) se puede volver a iniciar si el error se ha corregido. En particular, si los programas de aplicación se ejecutan de forma redundante para poder realizar una votación posterior de los datos variables generados, el funcionamiento de una aplicación ferroviaria, por ejemplo, puede continuar ejecutándose mientras que los individuales

Programas de aplicación o los ordenadores host se deben reiniciar.

Según una realización de la invención, se proporciona que

· Los conjuntos de datos de la aplicación que contienen secciones de datos que causan errores se actualizan utilizando las secciones de datos libres de errores de los conjuntos de datos de la aplicación correspondientes.
· los registros de datos de aplicación actualizados se incluyen nuevamente en el procesamiento posterior.

Por lo tanto, para la actualización se utilizan los correspondientes conjuntos de datos de la aplicación, que contienen las correspondientes secciones de datos sin errores. Esto significa que en los juegos de datos de aplicación se seleccionan las secciones de datos con datos de aplicación idénticos, es decir, registros de datos de aplicación que pertenecen al mismo grupo de juegos de datos de aplicación. Como ya se ha explicado, estos se diferencian sólo en las secciones de datos de prueba, pero no en las secciones de datos que representan los datos de la aplicación, es decir, secciones de datos de la aplicación, de modo que con ellas se puede utilizar, por así decirlo, para reparar el error detectado, siempre que las secciones de datos libres de error podrían identificarse mediante *voting*.

La actualización de las secciones de datos defectuosas permite que el entorno informático continúe funcionando continuamente. De lo contrario, cada vez se bloquearían más conjuntos de datos de aplicaciones a medida que se descubrieran errores, lo que significaría que el proceso ya no podría ejecutarse a largo plazo. En algún momento tampoco se podría realizar *voting* debido a una falta de redundancia en la generación de datos.

Además de la *voting*, la redundancia de los datos también permite el funcionamiento del entorno informático, es decir, el procesamiento de los conjuntos de datos de la aplicación, mientras se identifican y corrigen los

errores en los conjuntos de datos de la aplicación individuales. Estos últimos sólo quedan excluidos del proceso hasta que hayan sido reparados mediante el procedimiento anterior. Sin embargo, aunque se excluyen los conjuntos de datos de aplicaciones corruptos, existen conjuntos de datos de aplicaciones redundantes adicionales con datos intactos con los que se puede seguir llevando a cabo el método sin provocar retrasos en la aplicación. Los conjuntos de datos de aplicación reparados se pueden activar más tarde, por ejemplo, durante una interrupción del funcionamiento.

Además, se reivindica un producto de programa de ordenador con instrucciones de programa para la realización del método según la invención y/o sus ejemplos de realización, en el que el método según la invención y/o sus ejemplos de realización se pueden llevar a cabo mediante el producto de programa de ordenador.

Además, se reivindica un dispositivo de provisión para almacenar y/o proporcionar el producto de programa informático. El dispositivo de suministro es, por ejemplo, una unidad de almacenamiento que almacena y/o proporciona el producto del programa informático. De forma alternativa y/o adicional, el dispositivo de suministro es, por ejemplo, un servicio de red, un sistema informático, un sistema de servidor, en particular un sistema informático distribuido, por ejemplo, basado en *cloud* y/o un sistema informático virtual, que almacena y/o preferentemente pone a disposición el producto del programa informático en forma de un flujo de datos.

La puesta a disposición se realiza en forma de un bloque de datos de programa como archivo, en particular como archivo de *download*, o como flujo de datos, en particular como flujo de datos de *download*, del producto de programa de ordenador. Esta disposición también puede realizarse, por ejemplo, como una descarga parcial compuesta por varias partes. Un producto de programa informático de este tipo se lee en un sistema, por ejemplo, mediante el dispositivo de suministro, de modo que el procedimiento según la invención se lleva a cabo en un ordenador.

Más detalles de la invención se describen a continuación con referencia al dibujo. Los elementos de dibujo iguales o correspondientes están provistos cada uno de los mismos números de referencia y sólo se explican varias veces en la medida en que existen diferencias entre las figuras individuales.

Los ejemplos de realización que se explican a continuación son realizaciones preferidas de la invención. En los ejemplos de realización, los componentes descritos de las realizaciones representan cada uno de ellos características individuales de la invención que pueden considerarse independientemente unas de otras, que también desarrollan la invención independientemente entre sí y, por lo tanto, deben considerarse como parte de la invención individualmente o en conjunto. una combinación distinta a la mostrada. Además, los componentes descritos también se pueden combinar con las características de la invención descritas anteriormente.

Se muestra:

La figura 1 muestra una aplicación ferroviaria con un entorno informático con sus relaciones causales, pudiendo realizarse un ejemplo de realización del procedimiento según la invención con el entorno informático.

La figura 2 (que consta de las figuras parciales 2A y 2B, distribuidas en dos páginas) muestra una realización ejemplar del método según la invención utilizando un entorno informático en la aplicación ferroviaria según la figura 1 con dos ordenadores *host* como diagrama de bloques, donde las unidades funcionales individuales contienen módulos de programa que dan como resultado programas de aplicación y cada uno de ellos puede ejecutarse en uno o más procesadores y las interfaces pueden diseñarse correspondientemente mediante tecnología de *software* o *hardware*.

Las figuras 3 y 4 muestran realizaciones ejemplares del método según la invención como diagrama de flujo, donde los pasos individuales del método se pueden implementar individualmente o en grupos mediante módulos de programa y donde las unidades funcionales e interfaces según la figura 2 se indican a través de ejemplo.

La figura 5 muestra una posible secuencia de proceso para el método para la operación asistida por ordenador de una unidad de almacenamiento y para la ejecución asistida por ordenador de un programa de aplicación.

En la figura 1 se muestra esquemáticamente una aplicación ferroviaria que está controlada por un entorno informático RU. La aplicación ferroviaria dispone de carriles GL, en los que se muestran a modo de ejemplo distintos componentes de la tecnología de cajas de señales. Este es un motor de aguja WA que puede configurar una aguja WH. Además, en una de las vías está instalado una baliza BL, con la que se puede intercambiar información con los trenes que pasan por la baliza. Finalmente, se muestra una señal luminosa LS, que está controlada por un controlador para señal luminosa CL. El entorno informático RU puede tener varios ordenadores *host* HR1, HR2, HR3, a los que se distribuyen aplicaciones para controlar la aplicación ferroviaria en forma de programas de aplicación (más sobre esto a continuación). El primer ordenador *host* HR1 lo proporciona un centro de datos RZ y está conectado a una primera unidad de almacenamiento SE1 a través de una primera interfaz S1. El centro de datos RZ puede ser gestionado, por ejemplo, por un proveedor de

servicios del operador ferroviario o por el propio operador ferroviario. El primer ordenador *host* HR1 está conectado a una *cloud* CLD a través de una segunda interfaz S2, de modo que no está vinculado localmente a la aplicación ferroviaria.

En un centro de control LZ del operador ferroviario se aloja el segundo ordenador *host* HR2, que también está conectado a través de una tercera interfaz S3 con la *cloud* CLD. Además, el segundo ordenador *host* HR2 está conectado a una segunda unidad de almacenamiento SE2 a través de una cuarta interfaz S4.

El entorno informático RU presenta también, por ejemplo, un posicionador STW, en el que se aloja el tercer ordenador *host* HR3, que está conectado a través de una sexta interfaz S6 con una tercera unidad de almacenamiento SE3. Además, el tercer ordenador *host* HR3 tiene una quinta interfaz S5 con el segundo ordenador *host* HR2. El ordenador *host* HR3 también podría conectarse a la *cloud* CLD de una manera no mostrada. El tercer ordenador *host* HR3 presenta además una séptima interfaz S7 con el motor de aguja WA, una octava interfaz S8 con el controlador para señal luminosa CL y una novena interfaz S9 con la baliza BL.

Todas las interfaces S1 ... S9 según la figura 1 están básicamente cableadas o también pueden implementarse mediante tecnología de transmisión inalámbrica, por ejemplo, radio. La disposición de los ordenadores *host* HR1 ... HR3 es sólo un ejemplo y puede ampliarse según se desee para sistemas ferroviarios más complejos. Un entorno informático se define porque los ordenadores *host* HR1 ... HR3 implicados pueden comunicarse entre sí y, por lo tanto, las aplicaciones se pueden procesar a través de los ordenadores *host* HR1 ... HR3, teniendo en cuenta las capacidades informáticas disponibles. Para ello se crean instancias de cálculo no representadas, que se describirán más detalladamente a continuación (compárese con la figura 2, donde están representadas la instancias informática redundante RP1 ... RPn).

La figura 2 muestra un ejemplo de la configuración del primer ordenador *host* HR1 y del segundo ordenador *host* HR2 según la figura 1. La integración de otros ordenadores *host* se puede realizar de forma análoga. Los ordenadores *host* están organizados de tal manera que se organiza en los ordenadores *host* el dominio de determinados complejos de tareas en forma de complejos de programas PK1, PK2, PK3, PK4, que se componen de programas de aplicación individuales AP1 ... AP5.

Los complejos de programas generalmente combinan una serie de programas de aplicación cuyo procesamiento conjunto se puede resumir con vistas a la totalidad de las aplicaciones. En particular puede estar previsto que todos los programas de aplicación contenidos en un conjunto de datos se combinen en un complejo de programas. Para ello se tiene en cuenta que el juego de datos genera una combinación de secciones de datos en función de los datos utilizados, mientras que paralelamente un complejo de programas combina los correspondientes programas de aplicación a los que están asignadas las secciones de datos.

Por datos de configuración KD1 ... KD13 se deben entender datos que configuran programas de aplicación para las necesidades individuales de la presente aplicación. La configuración define la interacción entre diferentes programas de aplicación, así como la función de los programas de aplicación en los componentes de *hardware* en los que están instalados. La configuración también contiene ajustes al caso de uso actual para el cual está destinado el programa de aplicación en cuestión (por ejemplo, parámetros que pueden diferir en diferentes casos de uso).

El quinto programa de aplicación AP5 muestra también que esto se puede realizar mediante subprogramas individuales. Los subprogramas del quinto programa de aplicación AP5 son un *gateway* GW, un *voter* VT, un *generador de impulsos* TG y un *message broker* MB (más sobre esto a continuación). Sin embargo, esto sólo debe entenderse como un ejemplo. Alternativamente, por ejemplo, el generador de reloj podría ejecutarse en otra aplicación (no segura), mientras que los subprogramas restantes se ejecutan como se describe en el programa de aplicación AP5 (seguro).

Los programas parciales en el sentido de la invención son generalmente unidades más pequeñas, como por ejemplo módulos de programa, que en su conjunto forman el programa de aplicación. Por lo tanto, es ventajoso poder construir programas de aplicación de forma modular, es decir, por ejemplo, prever módulos de programa que se utilicen en varios programas de aplicación. Los programas parciales se pueden configurar con diferentes datos de configuración en función de su uso. Por lo tanto, los programas parciales permiten crear programas de aplicación más fácilmente y así adaptar más fácilmente el entorno informático a una aplicación.

En relación con la creación de complejos de programas, programas de aplicación y programas parciales, cabe señalar que, a los complejos de programas, a los programas de aplicación y a los programas parciales se les pueden asignar datos de configuración. Esto puede dar lugar a que datos específicos con el mismo contenido se guarden varias veces, dando lugar cada una a secciones de datos que a su vez pueden asignarse claramente a un complejo de programas, programas de aplicación o programas parciales. Lo importante aquí es la posibilidad de una asignación clara para tener disponibles secciones de datos claramente direccionables para implementar funciones de prueba.

El quinto programa de aplicación AP5 está organizado de la misma manera en todos los complejos de programas PK1 ... PK4. Los mensajes se pueden intercambiar con Cloud CLD a través del *Gateway* GW. El

Gateway GW forma por tanto las interfaces S2 y S3 según la figura 1. Los mensajes dentro del complejo de programas PK1 ... PK4 se distribuyen a través del *Message Broker* MB, preferentemente mediante el método de publicación-suscripción. *El gateway* GW, por ejemplo, utiliza una interfaz S14 para poner a disposición de las instancias informáticas redundantes RP1 ... RPn los mensajes recibidos a través del *Message Broker* MB. A continuación, acceden a ellas las instancias informáticas redundantes RP1 ... RPn. Esto se indica mediante los nodos KN, que están indicados en la interfaz S14 (y también en las otras interfaces S10 ... S13 descritas a continuación).

En la figura 2, para mayor claridad, los complejos de programas PK1 ... PK4 están completamente implementados en un ordenador *host* HR1 ... HR2. En realidad, los complejos de programas con sus programas de aplicación AP1 ... AP5 y sus subprogramas también pueden ejecutarse distribuidos en varios ordenadores *host* (no representados). Ventajosamente, esto permite aprovechar las capacidades de los ordenadores *host,* si éstos no ofrecen suficiente capacidad para configurar un complejo de programas completo, compartiendo la capacidad de varios ordenadores *host* para el complejo de programas en cuestión.

Los complejos de programas pueden, por ejemplo, estar destinados a un conjunto específico de tareas. Por ejemplo, se puede utilizar un complejo de programas para controlar un componente ferroviario específico (señal, posicionador, aguja, contador de ejes, etc.). En general, se requieren varios programas de aplicación para controlar este componente ferroviario. En particular, también se requiere el programa de aplicación AP5 ya explicado anteriormente para garantizar la ejecución segura de la aplicación y la comunicación con otros ordenadores host. Se trata de un programa de utilidad que sirve para garantizar el funcionamiento del ordenador host y, por tanto, ejecuta una aplicación relacionada con el ordenador host (a diferencia de los programas de aplicación AP1 ... AP4, que procesan datos de usuario para componentes ferroviarios, que por lo tanto se denominan programas de utilidad). El programa de aplicación AP5 también se ejecuta en al menos una instancia informática RP9... RP12 por ordenador host, pero preferiblemente no de forma redundante.

Por múltiples instancias informáticas redundantes en el sentido de la invención se entiende una implementación de software en los ordenadores *host* HR1 ... HR3, que permite el procesamiento paralelo, es decir simultáneo, de los programas de aplicación AP1 ... AP4, preferiblemente dentro del programa respectivo complejo PK1 ... PK4. En la figura 2 se muestran complejos de programas con dos instancias de cálculo redundantes, pero preferentemente se utilizan tres instancias de cálculo redundantes, aunque también son imaginables más instancias de cálculo redundantes RP1 ... RPn, como se muestra a modo de ejemplo para el primer complejo de programas. A continuación, se explica el procedimiento para procesar los programas de aplicación utilizando el primer complejo de programas PK1 para la primera instancia informática redundante RP1, la segunda instancia informática redundante RP2 ... y la enésima instancia informática redundante para procesar el primer programa de aplicación AP1. En los complejos de programas PK2 ... PK4 el procesamiento se realiza de forma correspondiente, por lo que no es necesario explicarlo por separado.

En la primera instancia informática redundante RP1, ... y en la enésima instancia informática redundante RPn se procesa el primer programa de aplicación AP1 de forma redundante, es decir, simultáneamente y en paralelo. Se trata de un programa de aplicación que asume una tarea para la aplicación ferroviaria según la figura 1. Desde la primera instancia informática redundante RP1 hasta la enésima instancia informática redundante RP2 también están disponibles los primeros datos de configuración KD1, que son necesarios para ejecutar el primer programa de aplicación AP1 para procesar la tarea individual de la aplicación ferroviaria. Por ejemplo, el primer programa de aplicación AP1 se puede utilizar en general para controlar señales luminosas, garantizando los primeros datos de configuración KD1 la aplicación del primer programa de aplicación AP1 a la señal luminosa LS según la figura 1. En este caso debe garantizarse, por ejemplo, la comunicación con el controlador CL según la figura 1.

Los datos de configuración KD1 ... KD13 también están disponibles para todos los demás complejos de programas PK1 ... PK4, programas de aplicación AP1 ... AP4 así como programas parciales MB, TG, VT, GW. Por consiguiente, los datos de configuración KD1 ... KD13 contienen los datos necesarios para que los complejos de programas, programas de aplicación y subprogramas puedan asumir las tareas que se les asignan en la aplicación correspondiente. Los datos de configuración no se pueden modificar y, por lo tanto, se pueden guardar en una sección de datos con un inicio y un final conocidos. Asimismo, todos los complejos de programas PK1 ... PK4, los programas de aplicación AP1 ... AP4 así como los programas parciales TG, VT, GW, MB se guardan como secciones de datos con un inicio y un final conocidos. Para ello están disponibles, por ejemplo, según la figura 1 la primera unidad de almacenamiento SE1, la segunda unidad de almacenamiento SE2 y la tercera unidad de almacenamiento SE3. Los datos que se almacenan en una de las unidades de almacenamiento mencionadas o permanecen almacenados en una de las unidades de almacenamiento mencionadas durante un cierto período de tiempo se someten a pruebas periódicas, mediante las cuales se pueden detectar errores de almacenamiento en los datos almacenados (más sobre esto a continuación). Los errores de almacenamiento son errores que ocurren o surgen en los datos cuando los datos se almacenan o recuperan mientras están almacenados en la unidad de almacenamiento.

Los datos que cambian mientras se procesan los programas se intercambian como mensajes entre los socios involucrados. Como ya se ha mencionado, para ello está disponible el *Message Broker* MB. Además, los

distintos ordenadores *host* HR1, HR2 se comunican entre sí a través de las interfaces externas S2, S3, por ejemplo, mediante una *cloud* CLD, de modo que también se pueden intercambiar datos entre diferentes complejos de programas PK1 ... PK4 de diferentes ordenadores *host*. Una vez modificados los datos, se almacenan de nuevo en la primera unidad de memoria SE1, en la segunda unidad de memoria SE2 o en la tercera unidad de memoria SE3. También pueden aparecer errores en los datos durante el procesamiento de los datos, que en el contexto de esta invención se denominan más precisamente errores de procesamiento.

Los procesos en la aplicación ferroviaria según la figura 1 son relevantes para la seguridad operativa de la aplicación ferroviaria. Esta es la razón por la que el primer programa de aplicación AP1 en la primera instancia informática redundante RP1 se procesa en paralelo en el tiempo, es decir, de forma redundante, hasta la enésima instancia informática redundante RPn. La primera instancia informática redundante RP1 y la segunda instancia informática redundante RP2 (etc.) envían el resultado al procesar la aplicación al *Message Broker* MB, concretamente la primera instancia informática redundante RP1 a través de la undécima interfaz S11 y la segunda instancia informática redundante a través de la interfaz S12 (etc.). Estos resultados se obtienen a través de las interfaces indicadas por el *voter* VT que realiza la votación. Sólo si la mayoría de los resultados coinciden (es decir, ambos resultados para dos instancias informáticas redundantes, al menos dos resultados para tres instancias informáticas redundantes, al menos tres resultados para cuatro instancias informáticas redundantes, ... al menos n/2+1 para pares y n/2+0,5 para n impar), el resultado se pone a disposición del *Message Broker* a través de la decimotercera interfaz S13 y puede ser recuperado por *el Gateway* GW a través de la decimotercera interfaz S13 para su transmisión a otras unidades a través de la segunda interfaz S2.

Para que los resultados del cálculo estén disponibles al mismo tiempo para *el voting* del *voter* VT, los procesos en la primera instancia informática redundante RP1 y en la segunda instancia informática redundante RP2 (etc.) se sincronizan a través del generador de impulsos TG. Esto proporciona señales del generador de impulsos a través de la décima interfaz S10, a la que también se puede acceder a través del *Message Broker* MB desde la primera instancia informática redundante RP1 y la segunda instancia informática redundante RP2.

El tipo de procesamiento de tareas descrito por el primer programa de aplicación AP1 y el segundo programa de aplicación AP2 está garantizado por el quinto programa de aplicación AP5. El quinto programa de aplicación AP5 es un programa de aplicación interno que soporta la funcionalidad de los ordenadores *host* HR1 ... HR3. Esto deja claro que los programas de aplicación deben estar disponibles no sólo para la aplicación de la aplicación ferroviaria según la figura 1 (programas de utilidad), sino también para el procesamiento de aplicaciones en los ordenadores *host* HR1 ... HR3 (programas de utilidad).

La combinación de programas de aplicación en complejos de programas y la división de programas de aplicación en programas parciales facilita la integración de programas de aplicación y la comprobación de que las tareas se procesan en busca de errores. Para este propósito, los datos se combinan en secciones de datos, cada una de las cuales puede identificarse y abordarse de manera única como tal (definiendo un comienzo de la sección de datos y un final de la sección de datos). Como ya se ha mencionado, los programas parciales, los programas de aplicación, los complejos de programas y los datos de configuración asociados se definen en secciones de datos (aunque éstas suelen constar de un gran número de secciones de datos). Los datos requeridos se guardan preferiblemente varias veces utilizando las características de diversidad para que cada sección de datos y archivos de configuración puedan asignarse de forma única. Es decir, en este caso no ocurre que diferentes programas de aplicación, si utilizan datos de configuración idénticos, accedan a la misma ubicación de almacenamiento para estos datos, sino que accedan siempre a la sección de datos que les ha sido asignada en la que están disponibles los datos.

En la figura 3, se muestra esquemáticamente la etapa del método de codificar datos por primera vez según la invención. Esto se muestra para un entorno informático, que consta, por ejemplo, del primer ordenador *host* HR1, la primera unidad de almacenamiento SE1 y las segundas unidades de almacenamiento SE2. En principio, el primer ordenador *host* HR1 puede acceder a datos que están almacenados en la primera unidad de almacenamiento SE1 y en la segunda unidad de almacenamiento SE2. Se puede leer RE y escribir WT, lo que se indica con las flechas correspondientes.

Por ejemplo, las aplicaciones AP para ejecutar programas de aplicación pueden almacenarse en las unidades de almacenamiento SE1, SE2. Además, es posible almacenar datos de conteo ZD así como una reserva VR de datos característicos de diversidad DD1 ... DD4. Para utilizar los datos de aplicación AD para aplicaciones en el sentido de la invención, es decir, para llevar a cabo un método para la ejecución asistida por ordenador de un programa de aplicación de la manera según la invención, es necesario almacenar los datos de aplicación AD. En forma de conjuntos de datos de aplicación ADS, que se crean y ejecutan según el procedimiento según la figura 3, se puede almacenar una codificación COD codificada.

La figura 3 muestra simplemente a modo de ejemplo cómo los datos de aplicación AD, los datos de conteo ZD y las características de diversidad DD1 ... DD4 ocupan áreas de memoria individuales de la primera unidad de memoria SE1 y de la segunda unidad de memoria SE2. En principio, es arbitrario dónde se almacenan los datos correspondientes, se obtienen mediante un direccionamiento adecuado y no existen restricciones en

cuanto a en qué conjunto de datos de aplicación ADS se almacena en las unidades de almacenamiento SE1, SE2.

Además, la diversidad de las características de diversidad DD1 ... DD4 se indica mediante un sombreado, con lo que se pretende dejar claro que los conjuntos de datos de aplicación ADS pueden caracterizarse por las características de reserva del VR original. Como se muestra en la reserva VR, las opciones disponibles son un sombreado longitudinal, un sombreado transversal y dos sombreados oblicuos, que forman un ángulo de 90° entre sí. El conjunto de datos de aplicación ADS mostrado en detalle en la FIG. 3 tiene, por ejemplo, el sombreado que indica las características de diversidad DD1.

Como muestra el conjunto de datos de aplicación ampliado ADS, éste consta de una sección de datos DA para los datos de aplicación AD y una sección de datos de prueba PA, que presenta los primeros datos característicos de diversidad DD1, un elemento de conteo en los datos de conteo ZD y datos de redundancia RD. Los primeros datos característicos de diversidad DD1, los datos de conteo ZD, los datos de redundancia RD, que se rellenan con un valor inicial, y los datos de aplicación AD se escriben, por ejemplo, en una memoria principal en el ordenador central HR1 y se combinan para formar el registro de datos de la aplicación ADS. A continuación, se codifica el conjunto de datos de aplicación ADS y se escribe en la primera unidad de almacenamiento SE1, indicándose también la diversidad basada en los primeros datos característicos de diversidad DD1 en la unidad de almacenamiento SE1 mediante el rayado mencionado. El registro de datos de la aplicación ADS está disponible aquí para su posterior procesamiento.

En la Figura 4, se muestra esquemáticamente el uso del conjunto de datos de aplicación según la Figura 3 y otros conjuntos de datos de aplicación con las segundas características de diversidad DD2 y las terceras características de diversidad DD3. Los tres conjuntos de datos de aplicaciones ADS mostrados deben contener datos de aplicaciones idénticos. La sección de datos de prueba PA está ocupada en cada caso con las diferentes características de diversidad DD1, DD2, DD3. Los datos de conteo ZD y los datos de redundancia RD también pueden diferir entre sí.

Los conjuntos de datos de aplicación ADS se ejecutan ahora en tres instancias informáticas RP1, RP2, RPn (en el juego principal explicado n es igual a 3, pero también podría tener un valor diferente). Para ello se leen los conjuntos de datos de aplicación ADS en la instancia informática (RE). A las instancias de cálculo RP1, RP2, RPn se le asigna a cada una una determinada diversidad, que se representa mediante un sombreado. Esta eclosión corresponde a las de las características de diversidad DD1 ... DD3 de la reserva VR.

También se puede observar que una instancia informática RPn también puede procesar datos de dos diversidades, en el presente caso los conjuntos de datos de aplicación ADS están marcados con las características de diversidad DD3 y con las características de diversidad DD4. De esta manera, se puede lograr una utilización óptima de la capacidad informática disponible a través de la instancia informática RPN.

Las instancias de cálculo RP1 ... RPn leen cada una de ellas los conjuntos de datos de aplicación ADS con la diversidad correcta. Esto se consigue mediante programas de utilidad que se ejecutan en segundo plano, de modo que las instancias informáticas RP1 ... RPn reciben automáticamente los conjuntos de datos de aplicación ADS correctos. Esto se muestra con más detalle para el conjunto de datos de aplicación ADS según la figura 3 con las características de diversidad DD1 y se explica con más detalle.

Si se recupera este registro de datos de aplicación ADS, primero se decodifica (DEC) y luego se lee (RE). Mediante la decodificación se pueden leer los primeros datos característicos de diversidad DD1, los datos de conteo actuales ZD y los datos de redundancia RD junto con los datos de aplicación AD y se pueden aplicar programas de utilidad que detecten cualquier error de memoria que pueda haberse producido. Se puede comprobar si las características de diversidad DD1 proceden de la reserva VR original y/o coinciden con la diversidad de la primera instancia informática RP1. Con los datos de conteo ZD se puede comprobar el desarrollo correcto de las ejecuciones de prueba, ya que estos deben identificar la ejecución de prueba que se está ejecutando actualmente o la ejecución de prueba que se ha realizado anteriormente. Sólo si la verificación muestra que los datos no tienen errores de memoria, se liberan para lectura RE y son procesados por la primera instancia informática RP1.

Después de procesar el registro de datos de aplicación ADS, la primera instancia informática RP1 lo vuelve a escribir en la primera unidad de almacenamiento SE1. También en este caso se puede realizar una comprobación de la sección de datos de prueba, de los primeros datos característicos de diversidad DD1, de los datos de conteo ZD y de los datos de redundancia RD para identificar posibles errores de procesamiento en el procesamiento del conjunto de datos de aplicación ADS. Además, los datos de conteo ZD se equiparan con el elemento de conteo que caracteriza la ejecución de prueba actualmente en curso. A continuación, el conjunto de datos de aplicación ADS se codifica (COD) y se escribe en la primera unidad de almacenamiento SE1 (WT).

Este procedimiento, aunque no se muestra en detalle, también se lleva a cabo para los otros conjuntos de datos de aplicación en las instancias informáticas RP2, RPn. Después del procesamiento exitoso de los juegos de

datos de aplicación ADS, se puede realizar una votación adicional para los datos de aplicación AD para determinar si los juegos de datos de aplicación AD fueron modificados de manera idéntica incluso después del procesamiento por las instancias de cálculo RP1 ... RPn. Si este no es el caso, esto indica un error de procesamiento. Con tres instancias de cálculo RP1 ... RPn también se puede realizar una votación por mayoría, de modo que los datos de la aplicación, en su mayoría idénticos, se utilizan para su posterior procesamiento, mientras que los datos de la aplicación que difieren de estos se bloquean para su posterior procesamiento.

La Figura 5 del Apéndice pretende explicar una posible secuencia de proceso para el método para la operación asistida por computadora de una unidad de almacenamiento y para la ejecución asistida por computadora de un programa de aplicación. Antes de iniciar el método, se lleva a cabo un paso de inicialización INI, que permite el acceso adecuado a la memoria de una unidad de memoria (no mostrada). Luego de iniciar el procedimiento, en el lado derecho se muestra el procedimiento para realizar ejecuciones de prueba de la unidad de almacenamiento y en el lado izquierdo la ejecución de los programas de aplicación. Estas subáreas pueden realizarse individualmente una tras otra o preferiblemente en paralelo y, por lo tanto, se muestran una al lado de la otra.

En primer lugar, se explicará el procedimiento para realizar la prueba. En una etapa de determinación de los datos de conteo DTM_ZD se determina un valor inicial para los datos de conteo. Estos datos de conteo se ponen a disposición del procedimiento para la ejecución de un programa de aplicación (lado izquierdo) en un paso de salida para datos de conteo ZD_OT, en caso necesario, a través de un paso de entrada para datos de conteo ZD_IN.

La ejecución de prueba real consta de procedimientos repetitivos que se llevan a cabo para todos los conjuntos de datos de la aplicación almacenados en la unidad de almacenamiento (que se muestra a la derecha en la Figura 5). Para cada juego de datos de aplicación con los elementos de conteo actuales de los datos de conteo se realiza lo siguiente: En un paso de decodificación para el juego de datos de aplicación DEC_ADS se decodifica el juego de datos de aplicación. En un paso de prueba para los datos de conteo TST_ZD se comprueba si el elemento de conteo corresponde a la ejecución de prueba que se está probando actualmente o a la última ejecución de prueba. En una ejecución de prueba para las características de diversidad TST_DD, se comprueba si el conjunto de datos de la aplicación tiene características de diversidad que corresponden a la reserva disponible VT de características de diversidad (véanse las figuras 3 y 4). En un paso de prueba para los datos de redundancia TST_RD se comprueba si los datos de redundancia tienen un valor esperado, en particular un valor predeterminado (opcional).

Si se han realizado todos los pasos de la prueba, ¿un paso de consulta sobre las desviaciones ¿DVG? Se comprobó si uno de los pasos de prueba descritos anteriormente producía desviaciones del resultado esperado. Si este es el caso, se genera un error en un paso de salida para el error ER_OT (más información a continuación). Si no es así, el juego de datos de aplicación comprobado se codifica nuevamente en un paso de codificación para el juego de datos de aplicación COD_ADS, codificando siempre en los datos de conteo con el elemento de conteo de la ejecución de prueba actual. Si se ha realizado la ejecución de prueba para todos los registros de datos de la aplicación, estos tendrán en los datos de conteo el elemento de conteo actual y los datos de conteo se pueden actualizar en un paso de actualización de los datos de conteo UPD_ZD para el programa de utilidad en ejecución. de modo que ahora contengan el elemento de conteo de la ejecución de prueba iniciada posteriormente.

En el procedimiento para la ejecución asistida por ordenador de un programa de aplicación (lado izquierdo de la figura 5), después del paso de entrada ya mencionado para los datos de conteo ZD_IN, se realiza repetidamente el paso de descodificación para el correspondiente juego de datos de aplicación DEC_ADS para todos los necesarios. conjuntos de datos de aplicación de la aplicación realizados por el programa de aplicación. A continuación se realiza, como ya se describió para la ejecución de prueba, un paso de prueba para los datos de conteo TST_ZD, un paso de prueba para los datos característicos de diversidad TSD_DD y un paso de prueba para los datos de redundancia TSD_RD (opcional).

Lo especial es que la aplicación se realiza en una instancia informática no representada con una determinada diversidad, de modo que las características de diversidad comprobadas en el paso de prueba para las características de diversidad TSD_DT deben corresponder exactamente con la diversidad de la instancia informática en la que se va a llevar a cabo el programa de aplicación.

Incluso al ejecutar el programa de aplicación, ¿el DVG? Se comprobó si en los pasos de prueba TST... se detectaron desviaciones del contenido esperado de los conjuntos de datos de la aplicación. Si este es el caso, como ya se ha explicado, se genera un error en el paso de emisión de errores ERR_OT. De lo contrario, la verificación de los juegos de datos de aplicación continúa hasta que se hayan verificado todos los juegos de datos de aplicación necesarios para el programa de aplicación. Sólo bajo esta condición se ejecutará el programa de aplicación en un paso de ejecución para el programa de aplicación RUN_APP.

La comprobación de los juegos de datos de aplicación se puede realizar preferentemente paso a paso para el programa de aplicación (no representado con mayor detalle en la figura 5). Esto significa que el procesamiento

del programa de aplicación se divide en pasos de procesamiento. En este sentido, se deben comprobar todos los registros de datos de aplicación necesarios para el programa de aplicación, que son necesarios para que el programa de aplicación lleve a cabo el siguiente paso. En el paso de ejecución del programa de aplicación RUN_APP se ejecuta a continuación el paso correspondiente del programa de aplicación. Para cada programa de aplicación existen entonces en la figura 5 varios bucles de recursión que, tras el paso de codificación descrito a continuación para el juego de datos de aplicación COD_ADS (en el lado izquierdo de la figura 5), conducen de nuevo al paso de entrada de los datos de conteo.

Después del paso de ejecución del programa de aplicación RUN_APP se comprueba si los registros de datos de aplicación existentes tras la ejecución del programa de aplicación son registros de datos de aplicación nuevos. Esta prueba (llamada paso de consulta ¿nuevo conjunto de datos de aplicación NW_ADS?) es necesaria para que a los nuevos conjuntos de datos de aplicación se les asignen datos de prueba en un paso de determinación para una sección de datos de prueba DTM_PA, lo que permite una prueba posterior del nuevo conjunto de datos de aplicación ADS en pasos adicionales del método presentado. En cualquier caso, el nuevo registro de datos de aplicación o los registros de datos de aplicación antiguos se codifican nuevamente en el paso de codificación para registros de datos de aplicación COD_ADS y se almacenan en la unidad de almacenamiento. A continuación, se puede ejecutar otro programa de aplicación o, como se ha descrito anteriormente, un paso adicional de un programa de aplicación en ejecución (repetición de ZD_IN, paso de entrada para contar datos y pasos siguientes).

También se pueden procesar varios programas de aplicación simultáneamente en instancias informáticas redundantes. En este caso, el proceso para ejecutar programas de aplicación se mostraría varias veces en paralelo de una manera no mostrada.

El paso de ejecución del programa de aplicación RUN_APP también puede producir resultados que deberían generarse. Antes de que esto suceda, opcionalmente también se puede ejecutar un procedimiento de prueba, como se muestra en el centro de la figura 5. Aquí se llevan a cabo los pasos ya explicados, es decir, el paso de prueba para los datos de conteo TSD_ZD, el paso de prueba para los datos característicos de diversidad TSD_DD y (opcionalmente) el paso de prueba para los datos de redundancia TSD_RD. ¿En un paso de consulta posterior sobre las desviaciones DVG? Se vuelve a comprobar si se pudieron detectar desviaciones en los pasos de prueba. Si este es el caso, como ya se ha descrito, se genera una señal de error en un paso de salida para un error ERR_OT. De lo contrario, el resultado se emite en un paso de salida para el resultado OT_RS y/o se procesa en pasos posteriores.

Si en un paso de salida se emite un error ERR_OT, en el ejemplo de realización según la figura 5 el procedimiento se detiene inmediatamente. El entorno informático puede entonces restablecerse por medio del paso de inicialización INI y, por ejemplo, reiniciar el proceso.

Lista de símbolos de referencia

LZ	Centro de control
STW	Posicionador
RZ	Centro de datos
GL	Carril
WH	Aguja
WA	Motor de aguja
LS	Señal luminosa
CL	Controlador para señal luminosa
BL	Baliza

RU	Entorno informático
HR1 ... HR3	Servidor principal
SE1 ... SE3	Unidad de almacenamiento
S1 ... S14	Interfaz
CLD	Nube
RP1 ... RP8, RPn	Instancia informática redundante
AP1 ... AP5	Programa de aplicación
PK1 ... PK4	Complejo de programas
KD1 ... KD13	Datos de configuración
MB	*Message Broker*
TG	Generador de impulsos
VT	*Voter*
GW	*Gateway*
KN	Nodo

DD1 ... DD4	Características de diversidad
ZD	Datos de conteo

RD	Datos de redundancia
AD	Datos de la aplicación
DA	Sección de datos con datos de la aplicación
PA	Sección de datos de prueba
ADS	Conjunto de datos de la aplicación ADS
CDS	Conjunto de datos de aplicación codificados
GR	Grupo
VR	Reserva
COD	Codificación
DEC	Decodificación
RE	Lectura
WT	Escritura

DTM_ZD	Paso de determinación para conteo de datos
UPD_ZD	Paso de actualización de datos de recuento
DEC_ADS	Paso de decodificación para el conjunto de datos de la aplicación
COD_ADS	Paso de codificación para el conjunto de datos de la aplicación
TST_ZD	Comprobación para conteo de datos
TST_DD	Paso de prueba para características de diversidad
TST_RD	Paso de verificación para datos de redundancia
DVG?	Paso de consulta para desviaciones
ERR_OT	Paso de salida para errores
INI	Paso de inicialización
RUN_APP	Paso de ejecución del programa de aplicación
ZD_IN	Paso de entrada para conteo de datos
ZD_OT	Paso de salida para conteo de datos
OT_RS	Paso de salida para el resultado
NW_ADS?	Paso de consulta para un nuevo registro de datos de la aplicación
DTM_PA	Paso de definición

REIVINDICACIONES

1. Método para el funcionamiento asistido por ordenador de una unidad de almacenamiento, en el que

• Los datos se almacenan en la unidad de almacenamiento y los datos se codifican antes del almacenamiento,
• Los datos se recuperan de la unidad de almacenamiento y, después de la recuperación, se decodifican,

en donde la unidad de almacenamiento se monitorea para detectar errores usando una secuencia de tiempo de análisis asistido por ordenador realizándose pruebas de funcionamiento para la unidad de almacenamiento, caracterizado porque,
para la codificación inicial (COD) de los datos

• se genera o selecciona al menos un registro de datos de aplicación (ADS), que contiene secciones de datos con datos de aplicación para un programa de aplicación (AP1 ... AP5) y secciones de datos de prueba (PA),
• para cada registro de datos de aplicación (ADS), a la sección de datos de prueba (PA) se le asignan datos de conteo (ZD), que identifican la ejecución de prueba que se está llevando a cabo,
• cada registro de datos de aplicación (ADS) está codificado y almacenado, y eso para verificar los datos en la ejecución de prueba que se lleva a cabo después de recuperar y decodificar (DEC) los conjuntos de datos de la aplicación (ADS) respectivamente
• se detecta un error para un registro de datos de aplicación (ADS) si los datos de conteo (ZD) no identifican ni la ejecución de prueba que se está llevando a cabo actualmente ni la ejecución de prueba completada más recientemente,
• la sección de datos de prueba (PA) de la sección de datos de la aplicación correspondiente se llena con datos de conteo (ZD), que identifican la ejecución de prueba que se está llevando a cabo, si no se detectó ningún error,
• el registro de datos de aplicación (ADS) correspondiente se codifica y se almacena de nuevo si no se detecta ningún error.

2. Método según la reivindicación 1,
caracterizado porque,
antes de realizar la primera ejecución de prueba, se debe ajustar un valor inicial para los datos de conteo (ZD).

3. Procedimiento según una de las reivindicaciones anteriores,
caracterizado porque,
la prueba se ejecute al menos verificar todo al comienzo de la ejecución de la prueba para el procedimiento actual incluya conjuntos de datos de aplicaciones disponibles.

4. Procedimiento según una de las reivindicaciones anteriores,
caracterizado porque,
tan pronto como finalice una ejecución de prueba, a los datos de conteo (ZD) se les asigne un elemento de conteo que sigue al elemento de conteo para la asignación anterior de los datos de conteo (ZD).

5. Procedimiento para la ejecución asistida por ordenador de un programa de aplicación (AP1...AP5), en el que se opera una unidad de almacenamiento según una de las reivindicaciones anteriores y en el que se recuperan de la unidad de almacenamiento los conjuntos de datos de aplicación necesarios para la ejecución. y decodificado, mediante el cual

• después de decodificar (DEC) los registros de datos de la aplicación, se ejecuta el programa de aplicación (AP1 ... AP5),
• Al menos los conjuntos de datos de aplicación, que contienen secciones de datos modificadas o generadas mediante la ejecución del programa de aplicación (AP1 ... AP5), están codificados y almacenados en la unidad de almacenamiento.

6. Método según la reivindicación 5,
caracterizado porque,
al verificar los datos después de decodificar (DEC) los conjuntos de datos de la aplicación y antes de ejecutar el programa de aplicación (AP1 ... AP5) respectivamente

• se detecta un error para un juego de datos de aplicación (ADS) si los datos de conteo (ZD) no corresponden ni a la ejecución de prueba aun en curso ni a la ejecución de prueba completada más recientemente,
• el programa de aplicación (AP1 ... AP5) se ejecuta si no se detecta ningún error.

7. Procedimiento según una de las reivindicaciones anteriores,
caracterizado porque,
durante el funcionamiento asistido por ordenador de la unidad de almacenamiento para la codificación inicial (COD) de los datos adicionalmente

• al menos un grupo (GR) de conjuntos de datos de aplicación (ADS), que contiene secciones de datos con datos de aplicación (AD) idénticos para un programa de aplicación (AP1 ... AP5) y secciones de datos de prueba (PA), además, cada una con diferentes características de diversidad (DD1... DD4) son generados o seleccionados los datos característicos (DD1 ... DD4) de una reserva (VR) de posibles opciones de diversidad.
• cada registro de datos de aplicación (ADS) se codifica y almacena, y los datos se recuperan retomando y decodificando los registros de datos de aplicación (ADS), y los datos se almacenan codificando y almacenando los registros de datos de aplicación (ADS), y/o
que en la ejecución asistida por ordenador
el programa de aplicación (AP1...AP5) se ejecuta varias veces en instancias informáticas redundantes (RP1...RPn), recuperándose cada registro de datos de aplicación (ADS) de un grupo codificado (GR) de registros de datos de aplicación (ADS) de la unidad de almacenamiento para ejecutar los conjuntos de datos de aplicación requeridos (ADS) y se decodifica, donde
• para verificar los datos después de la decodificación (DEC) de los conjuntos de datos de aplicación (ADS), se detecta un error para un conjunto de datos de aplicación (ADS) si las características de diversidad (DD1 ... DD4) en el conjunto de datos de aplicación (ADS) son diferentes de todas las posibles características de diversidad (DD1 ... DD4) de esta instancia informática (RP1, RP2, RPn) difieren de la reserva (VR),
• se ejecuta el programa de aplicación (AP1 ... AP5) con las secciones de datos de cada registro de datos de la aplicación (ADS) del grupo relevante (GR) si no se detecta ningún error,
y/o
• al menos los registros de datos de aplicación (ADS), que contienen secciones de datos modificadas o generadas mediante la ejecución del programa de aplicación (AP1 ... AP5), se codifican del modo mencionado anteriormente y se almacenan en la unidad de almacenamiento.

8. Procedimiento según una de las reivindicaciones anteriores,
caracterizado porque,
antes de la codificación (COD), la sección de datos de prueba (PA) se crea de tal manera que tenga adicionalmente espacio de almacenamiento con datos de redundancia (RD) especificados.

9. Procedimiento según una de las reivindicaciones anteriores,
caracterizado porque,
las secciones de datos de la aplicación constan de palabras de datos.

10. Procedimiento según una de las reivindicaciones anteriores,
caracterizado porque
el método se lleve a cabo utilizando al menos dos procesadores o núcleos de procesador al mismo tiempo.

11. Método según la reivindicación 10,
caracterizado porque,
al menos dos procesadores o núcleos de procesador accedan a una misma unidad de memoria (SE1...SE3).

12. Procedimiento según una de las reivindicaciones anteriores,
caracterizado porque

• el *voting* se realiza con secciones de datos idénticas guardadas varias veces y/o con secciones de datos que han sido modificadas de manera idéntica varias veces.
• si se identifica un error debido al *voting*, los conjuntos de datos de la aplicación (ADS) contiene secciones de datos causales de la identificación del error pueden excluirse de un procesamiento posterior.

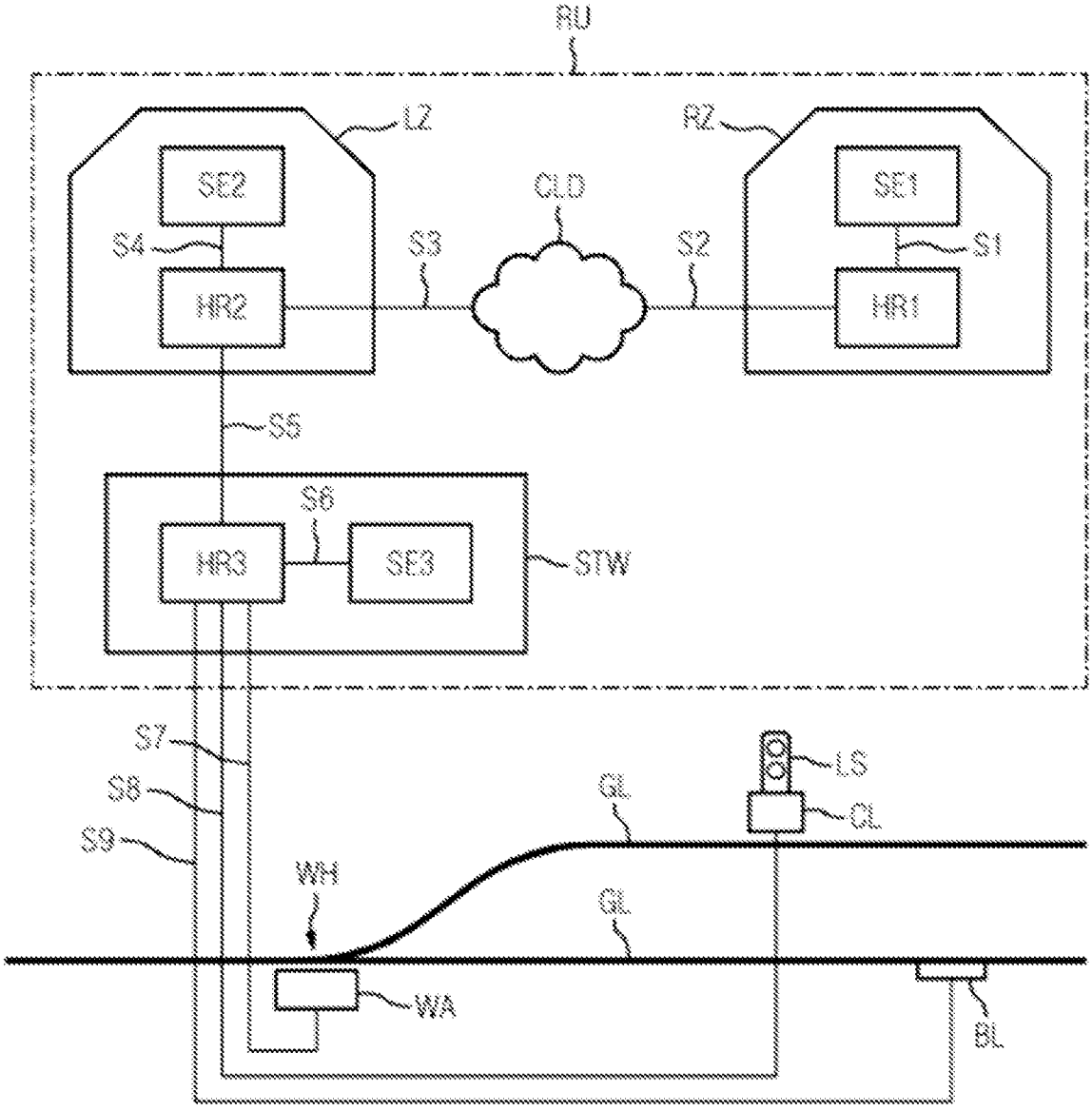
13. Procedimiento según una de las reivindicaciones anteriores,
caracterizado porque,

• Los conjuntos de datos de la aplicación (ADS), que contienen secciones de datos que causan errores, se actualizan utilizando las secciones de datos libres de errores de los conjuntos de datos de la aplicación (ADS) correspondientes.
• los registros de datos de aplicación (ADS) actualizados se incluyen nuevamente en el procesamiento posterior.

14. Producto de programa informático con instrucciones de programa para la realización del método según una de las reivindicaciones anteriores.

15. Dispositivo de suministro para el producto de programa informático según la reivindicación 14, en el que el dispositivo de suministro almacena y/o proporciona el producto de programa informático.

FIG 1


RU
LZ
RZ
SE2
SE1
CLD
S4
S3
S2
S1
HR2
HR1
S5
S6
HR3
SE3
STW
S7
S8
S9
LS
CL
QL
WH
GL
WA
BL

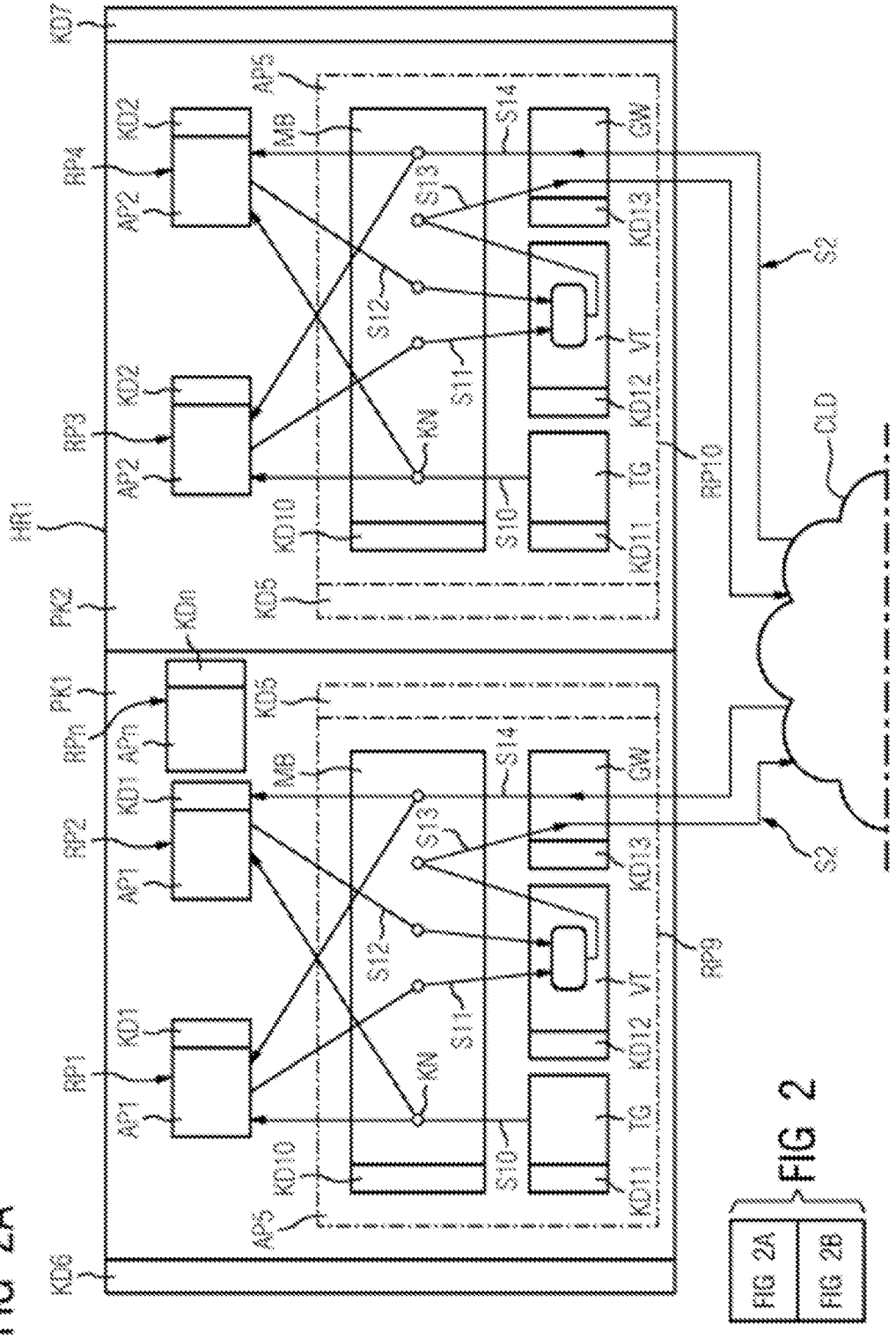

KD7
KD6
HR1
PK2
PK1
RP4
RP3
RP2
RP1
RPn
RP10
RP9
AP2
AP1
APn
APk
KD2
KD1
KDn
KD5
KD10
KD11
KD12
KD13
M6
S10
S11
S12
S13
S14
KN
VT
TG
GW
S2
CLD

FIG 2A

FIG 2

FIG 2A
FIG 2B

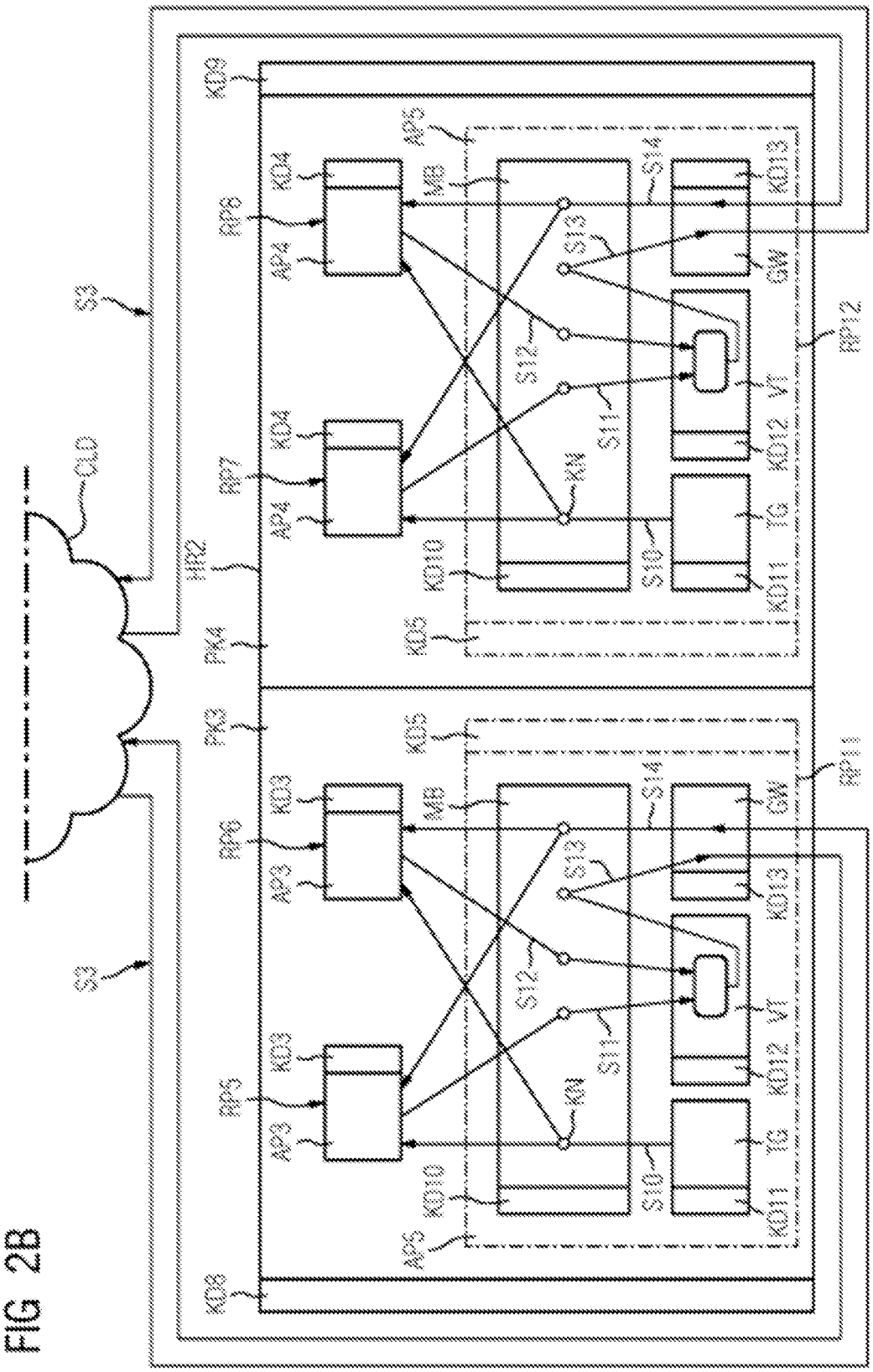

KD9
AP5
KD4
MB
S14
KD13
RP6
AP4
S13
GW
S12
RP12
VT
S11
KD12
RP7
KN
TG
S10
KD10
KD11
KD5
HR2
RK4
RK3
RP11
KD3
RP5
AP3
KD8
S3
CLD

FIG 2B

FIG 3

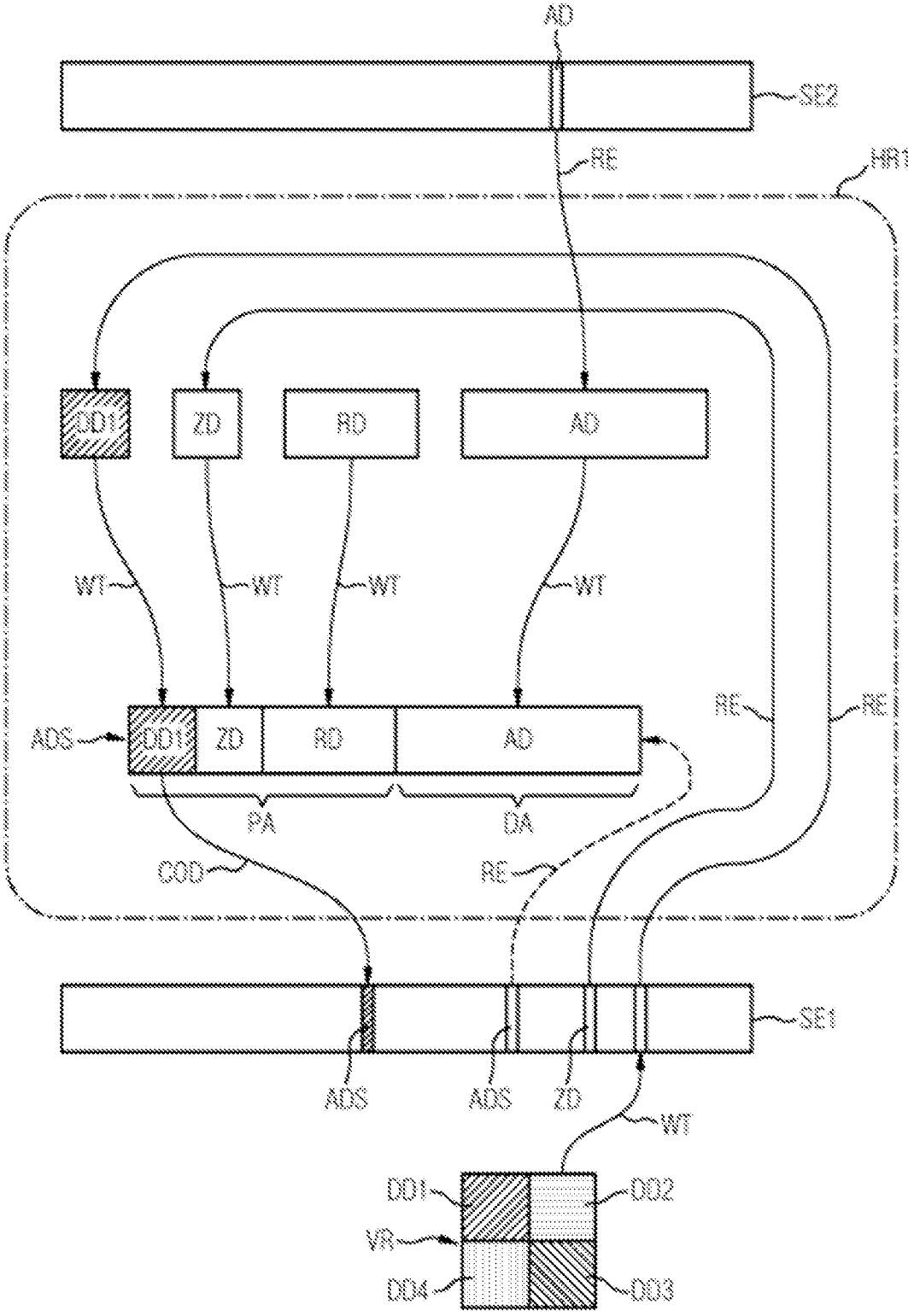

AD
SE2
RE
HR1
DD1
ZD
RD
AD
WT
WT
WT
WT
ADS
DD1
ZD
RD
AD
RE
RE
PA
DA
COD
RE
SE1
ADS
ADS
ZD
WT
DD1
DD2
VR
DD4
DD3

FIG 4

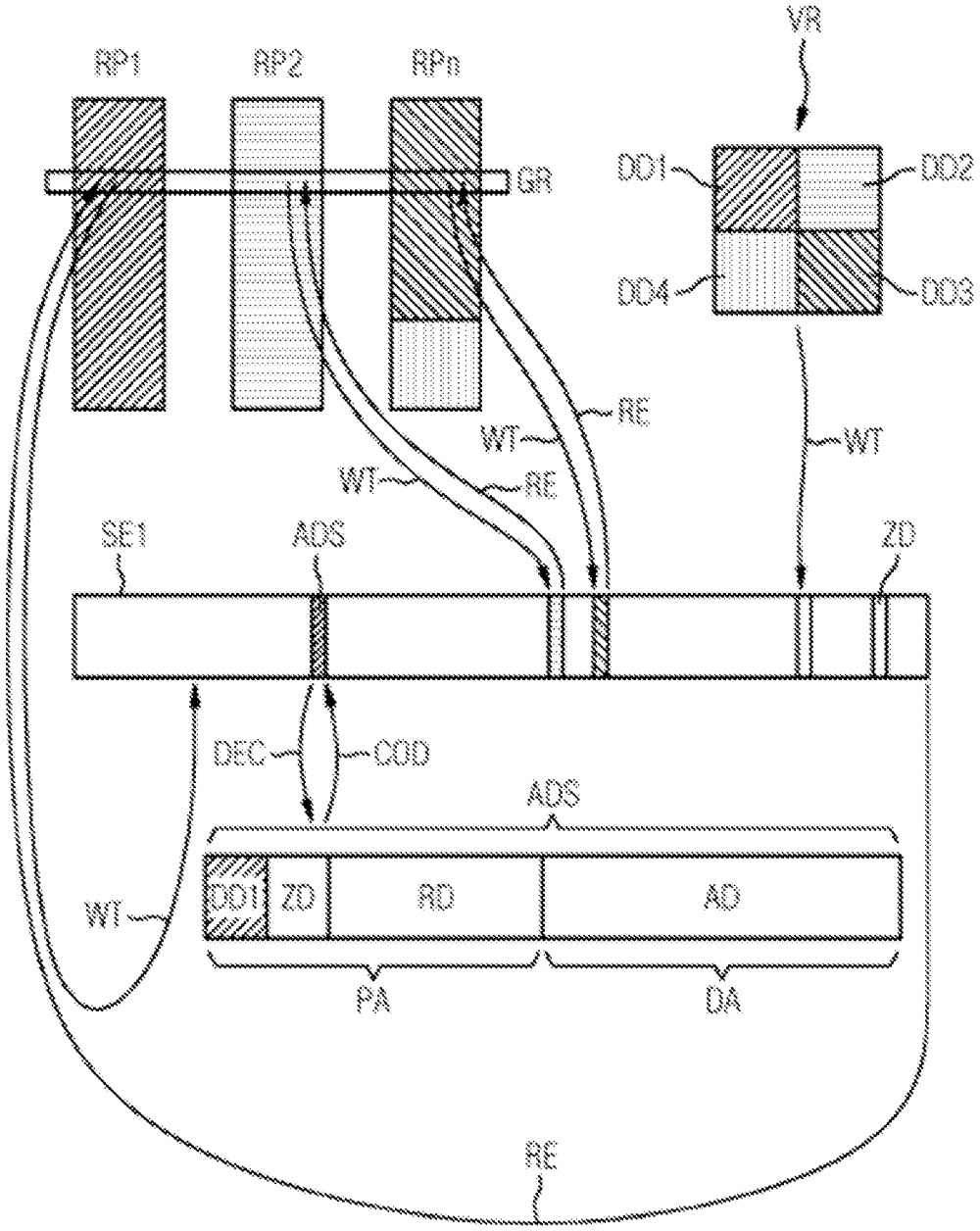
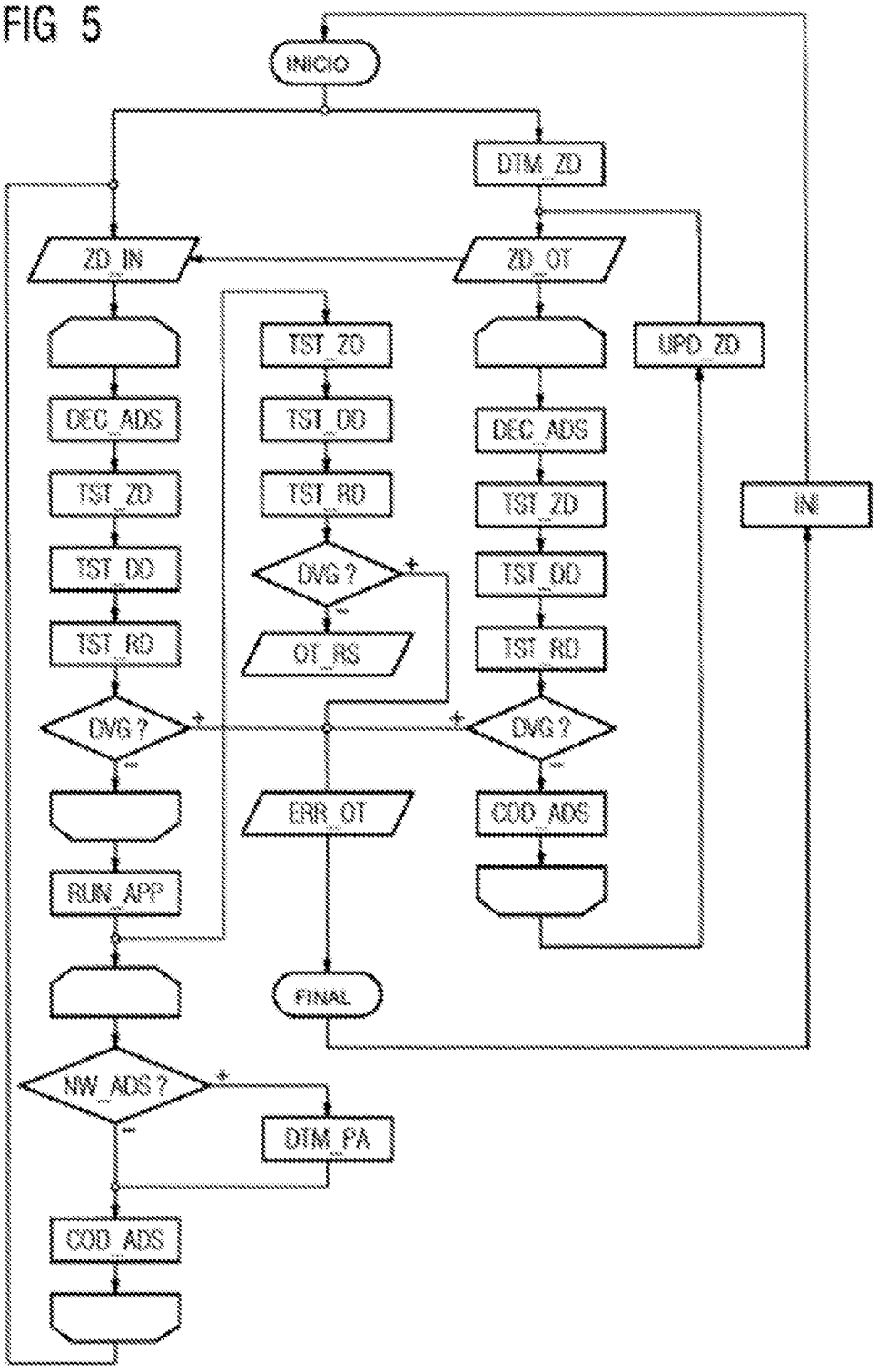

VR
RP1
RP2
RPn
GR
DC1
DC2
DC4
DC3
RE
WT
WT
RE
WT
SE1
ADS
ZD
DEC
COD
ADS
WT
DO1
ZD
RD
AD
PA
DA
RE

FIG 5


INICIO
DTM_ZD
ZD_IN
ZD_OT
DEC_ADS
TST_ZD
TST_DD
TST_RD
DVG ?
+
-
RUN_APP
NW_ADS ?
COD_ADS
TST_ZD
TST_DD
TST_RD
DVG ?
OT_RS
ERR_OT
FINAL
DTM_PA
DEC_ADS
TST_ZD
TST_DD
TST_RD
DVG ?
COD_ADS
UPD_ZD
IN