(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 681 522**

(51) Int. CI.:

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

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **18.07.2005 PCT/US2005/025250**

(87) Fecha y número de publicación internacional: **23.02.2006 WO06020094**

(96) Fecha de presentación y número de la solicitud europea: **18.07.2005 E 05772448 (6)**

(97) Fecha y número de publicación de la concesión europea: **13.06.2018 EP 1779245**

(54) Título: **Procedimiento y sistema para minimizar la pérdida en una aplicación informática**

(30) Prioridad:

20.07.2004 US 589262 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
13.09.2018

(73) Titular/es:

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

(72) Inventor/es:

SCHAEFER, STUART

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Procedimiento y sistema para minimizar la pérdida en una aplicación informática

Referencia cruzada a solicitudes relacionadas

La presente solicitud reclama prioridad a la Solicitud de Patente Provisional de los EE.UU, número 60/589.262, presentada el 20 de julio de 2004.

Campo técnico

La presente invención se refiere a la tolerancia general a fallos de las aplicaciones y los procedimientos y sistemas de software informáticos para hacer que las aplicaciones individuales puedan operar en escenarios de fallos para los cuales no fueron programadas o previstas.

Antecedentes

Los sistemas informáticos y las aplicaciones de software se han vuelto cada vez más complejos y distribuidos. Ambos factores contribuyen al problema común de la pérdida de datos. Cuando un usuario final opera una aplicación de software, normalmente salvaguardará los resultados de las operaciones en uno o más archivos de datos, en una base de datos o en otro lugar. La acción de cometer estas operaciones crea un cambio de estado en el sistema que puede actuar efectivamente como un punto de control. Los programadores de aplicaciones pasan una gran cantidad de tiempo asegurando que sus programas de software funcionen de la manera prevista en estos puntos de control, ya sea ejecutando o rechazando los cambios.

Sin embargo, también es común que se acumulen cambios de estado entre estos puntos de control. En la mayoría de las aplicaciones de software puede haber una cantidad apreciable de tiempo transcurrido u operaciones que se realizan entre ejecuciones. Si la aplicación falla durante este intervalo, las acciones tomadas por el usuario pueden perderse, con vuelta al último punto de control. El usuario debe volver a abrir la aplicación, estudiar su estado visualizable para comprender lo que se perdió y recrear aquellas acciones que se realizaron.

Los fallos de las aplicaciones pueden ocurrir por varias razones, incluyendo fallos de red, fallos de hardware, fallos sistémicos del servidor o del sistema y otros fallos operativos. En los nuevos modos de software en los que los usuarios desconectan ordenadores portátiles u otros dispositivos móviles de una red, o en los que las aplicaciones son transmitidas en continuo o se entregan en partes a los ordenadores cliente, las posibilidades de fallos aumentan. Muchas aplicaciones no están diseñadas para ser operadas mientras están desconectadas de la red, o para ser operadas sin la totalidad del programa y sus activos presentes en el tiempo de ejecución.

Es deseable proporcionar un medio para acomodar o superar estas y otras formas de fallos, eliminando el trabajo perdido tanto en los puntos de control como entre ellos sin requerir que las aplicaciones se vuelvan a escribir o teniendo en cuenta todas las formas de fallo, ya que esto sería costoso y prohibitivo en tiempo. Se propone un conjunto de procedimientos simples y de uso general para proporcionar la flexibilidad deseada sin modificación de cualquier aplicación de software, ni acceso al código o diseño de la aplicación de software.

Se han propuesto procedimientos para lograr soluciones a este problema para aplicaciones o propósitos específicos en el momento del diseño, tal como los procedimientos que se describen en la Patente de los EE.UU, número 6.014.681 a Walker, et al., titulada *Procedimiento para salvaguardar un documento utilizando una cadena de archivar en segundo plano.* La Patente de los EE.UU, número 5.748.882 a Huangin titulada *Aparato y procedimiento para computación tolerante a fallos* desvela bibliotecas de rutinas tolerantes a fallos que se han creado para que los desarrolladores de aplicaciones las usen. El procedimiento de la presente invención supera la limitación de requerir que una aplicación se diseñe y construya para que sea tolerante a fallos utilizando tales bibliotecas. La invención proporciona un marco de solución extensible para manejar los muchos requisitos de sistemas de software pasados, presentes y futuros.

El documento US B1 6 708 291 se refiere a sistemas informáticos y procedimientos de procesamiento de datos en los que los descriptores jerárquicos definen los niveles de gestión de fallos con el fin de monitorizar de manera inteligente el hardware y el software y actuar de manera proactiva de acuerdo con una política de fallos definida. También se describe en el contexto de un sistema informático, tal como un conmutador de red para su uso en Internet, WAN o LAN. También se establece que la arquitectura de software modular puede ser implementada en cualquier dispositivo de red u otros tipos de sistemas informáticos y no está restringida a un dispositivo de red.

El documento US 2002/133738 A1 se relaciona con sistemas de multitareas preventivas y un programa fallado, en el que se invoca un programa de descongelación cuando se detecta un programa en colisión. Por ejemplo, las llamadas al sistema operativo realizadas por un programa en colisión son monitorizadas y un grupo seleccionado de llamadas al sistema operativo son interceptadas antes de que sean ejecutadas por el sistema operativo. Además, un subconjunto del grupo seleccionado de llamadas al sistema operativo interceptadas es registrado en una memoria.

El documento US A 5 274 813 se refiere a un procedimiento y sistema que proporciona un sistema operativo para un ordenador, que puede volver a ejecutar un trabajo desde un punto interrumpido, incluso después de agotar por adelantado los recursos asignados al trabajo. El sistema comprende medios para monitorizar una posición de ejecución de un programa de usuario, y medios para monitorizar un valor de un recurso usado.

El documento EP A 0 433 979 se refiere a la provisión de un sistema informático de alta fiabilidad en el que el rendimiento, medido en fiabilidad así como en velocidad y compatibilidad de software, es capaz de ejecutar un sistema operativo que utiliza gestión de memoria virtual y que es capaz de detectar componentes del sistema de fallos y ponerlos fuera de línea. Además, el sistema también debería proporcionar una operación mejorada en situaciones de fallo de energía.

Sumario

El objeto de la invención es proporcionar un procedimiento y sistema con un mecanismo mejorado de detección de fallos para minimizar la pérdida de datos mientras se usa una aplicación informática.

Este objeto se resuelve mediante la presente invención como se reivindica en las reivindicaciones independientes.

Las realizaciones preferidas están definidas por las reivindicaciones dependientes.

La presente invención proporciona un procedimiento y sistema para prevenir o minimizar la pérdida de datos cuando se produce un fallo o un fallo potencial en una aplicación informática. El procedimiento y el sistema de la presente invención pueden minimizar o evitar la pérdida de datos sin requerir ninguna modificación o acceso al código para la aplicación que está en uso.

En la presente invención, se proporciona una aplicación de asistencia para monitorizar una aplicación primaria que se ejecuta en un ordenador objetivo para detectar la presencia de un fallo o un fallo potencial en la aplicación primaria. Cuando se detecta un fallo, la aplicación de asistencia selecciona un procedimiento para manejar el fallo y a continuación realiza el procedimiento seleccionado.

La aplicación de asistencia monitoriza la aplicación primaria interceptando llamadas a la aplicación primaria e inspeccionando los resultados de las llamadas. La aplicación de asistencia monitoriza las operaciones internas de la aplicación (tales como fallos lógicos) y / o operaciones externas (tales como pérdida de conectividad de red, fallo de los servicios externos requeridos por la aplicación, fallo de los dispositivos conectados y fallo de las redes de pares o servidores).

Si se identifica un fallo, la aplicación de asistencia determina qué controlador de fallos es el más apropiado. En la realización preferida de la presente invención, hay disposiciones para controladores de fallos que incluyen un controlador de redirección, un controlador de suspensiones y un controlador de instantáneas. El sistema es extensible para proporcionar la capacidad de utilizar cualquier cantidad de controladores diferentes para diferentes propósitos. Ejemplos de otros controladores pueden ser un controlador de reinicio automático o un controlador de rebobinado.

Si se selecciona el controlador de redirección, el controlador de redirección invoca una lógica de salvaguarda de la aplicación primaria. La lógica de salvaguarda puede ser invocada (a) llamando directamente al punto de entrada de salvaguarda de la aplicación primaria, (b) reiniciando el contador del programa al punto de entrada salvaguarda, o (c) insertando instrucciones de salto para forzar la llamada de salvaguarda.

Si se selecciona el controlador de suspensiones, el controlador de suspensiones suspende la aplicación y espera la notificación de que la reanudación puede ocurrir. Una vez que recibe esta notificación, el controlador de suspensiones (a) reanuda la aplicación primaria o (b) restablece el contador del programa de la aplicación primaria o (c) vuelve a invocar la API originalmente defectuosa. El controlador de suspensiones evita la reanudación de la dispersión de la aplicación primaria mientras la aplicación primaria está suspendida al hacer que la aplicación primaria solo se pueda controlar a través de medios predeterminados. La aplicación primaria puede ser suspendida durante la duración del fallo o puede ser suspendida por una interfaz de usuario que se anticipa a un estado de fallo.

Si se selecciona el controlador de instantáneas, el controlador de instantáneas suspende los procesos y subprocesos de la aplicación primaria en respuesta a la presencia de un fallo o un posible fallo en la aplicación primaria. El controlador de instantáneas toma entonces una instantánea de la memoria de la aplicación primaria, almacena la memoria en un almacenamiento no volátil, finaliza la aplicación primaria y, cuando se invoca, restaura la aplicación primaria. El controlador de instantáneas evita que se vuelva a invocar la aplicación primaria hasta que el fallo sea borrado o el usuario lo confirme explícitamente.

En las realizaciones que se han descrito más arriba, la aplicación de asistencia clasifica qué conjunto de fallos manejará, así como qué controladores puede invocar. El sistema puede obtener este conocimiento fuera de línea o en línea por medio de una variedad de técnicas. En el sistema más básico, el controlador simplemente se registrará para detectar todos los fallos.

De acuerdo con otro aspecto de la invención, se realiza un paso adicional para realizar un análisis estático en el código de aplicación primaria para determinar tanto su composición como su potencial para fallos, así como sus puntos de entrada de programas comunes tales como "salvaguardar" o "salir". Después de completar el análisis, el analizador puede presentar una lista de los fallos que son recuperables y los que no son recuperables y las recomendaciones sobre el manejo. Para ambos conjuntos de fallos, las acciones pueden ser proscritas para configurar los controladores de fallos. Para aquellos fallos que no son recuperables, el controlador de redirección se usa comúnmente. Si no existe un código de "salvaguardar" o es identificable, el sistema puede configurarse para simplemente salir fácilmente, aunque los datos no se conservarán necesariamente.

Para analizar los puntos de entrada del programa, la aplicación de asistencia puede usar el conocimiento de la plataforma para descubrir las fuentes. En una realización alternativa, los fallos y los puntos de entrada pueden ser identificados usando análisis dinámico. Por lo tanto, se puede rastrear el programa durante su ejecución para identificar estas firmas.

Breve descripción de los dibujos

La figura 1 muestra un diagrama esquemático de una realización del sistema y procedimiento de la presente invención;

la figura 2 muestra un diagrama esquemático de los procesos de detección de fallos utilizados en el sistema y el procedimiento de la presente invención que se muestra en la figura 1;

la figura 3 muestra un diagrama esquemático del funcionamiento de los contadores de programa en el sistema y procedimiento de la presente invención que se muestra en la figura 1;

la figura 4 muestra un diagrama esquemático de un analizador de programa usado por el sistema y el procedimiento de la presente invención que se muestra en la figura 1; y

la figura 5 muestra un diagrama de flujo de las etapas realizadas por el procedimiento de la presente invención que se muestra en la figura 1 para detectar y manejar fallos.

Descripción detallada de las realizaciones preferidas

Una aplicación de software 100 se ejecuta en un ordenador objetivo 101 como se muestra en la figura 1. Una aplicación de asistencia 102 se ejecuta simultáneamente en el mismo ordenador, aunque también puede existir en todo o en parte como un programa de servicio en un ordenador alternativo 103. Esta aplicación de asistencia 102 puede residir como un proceso independiente, inyectado en el espacio de memoria de la aplicación primaria o como parte del sistema operativo del ordenador como un controlador del dispositivo o de otra manera. No es necesario recompilar o modificar de otro modo la aplicación primaria 100. La aplicación primaria 100 puede existir como un programa independiente o en comunicación con uno o más ordenadores de pares o servidores en una red.

Haciendo referencia a la figura 2, la aplicación de asistencia 102 incluye un detector de fallos 201, un controlador de fallos 203 y una Base de Conocimientos 203. A medida que se ejecuta la aplicación primaria 100, el detector de fallos 201 monitoriza la aplicación primaria 100 para intentar detectar la presencia de un fallo. La detección de fallos se realiza tanto en respuesta a las operaciones internas de la aplicación primaria 100, como en respuesta a otros factores externos, tales como la pérdida de conectividad de red, el fallo de los servicios externos requeridos por la aplicación, el fallo de dispositivos conectados como impresoras y módems, el fallo de los ordenadores de pares o servidores en su red. Para acomodar fallos internos, el detector de fallos intercepta varios mensajes y acciones tomados por la aplicación primaria 100. En un procedimiento de ejemplo (que se muestra en la figura 2), la aplicación de asistencia 102 monitoriza las rutinas de asignación de llamadas a memoria de la aplicación primaria, tales como el uso de malloc () y fallado () en la biblioteca de tiempo de ejecución de C estándar. Si la rutina malloc () no puede asignar la memoria que fue solicitada, se habrá producido un fallo de memoria.

En este ejemplo, la llamada de la aplicación primaria a la rutina malloc () es interceptada por el detector de fallos 201. Se hace notar que en una realización alternativa de la presente invención, el detector de fallos podría operar de manera que no intercepte realmente la llamada de rutina, sino monitorizar su salida para detectar fallos tales como excepciones, códigos de retorno incorrectos u otro comportamiento incorrecto. La llamada interceptada se reenvía a la rutina malloc (). En su retorno, el detector de fallos inspeccionará los resultados de la llamada. En este punto, el detector de fallos se bifurcará y devolverá el control a la aplicación primaria 100, o invocará un controlador de fallos 202. El controlador de fallos 202 puede contener decisiones codificadas estáticamente y / o un sistema tal como una Base de Conocimientos configurable (203) (como se muestra en la figura 2) para invocar una rutina de manejo de fallos basada en un conjunto más grande de comportamientos y heurística.

En un ejemplo, la Base de Conocimientos 203 contiene un conjunto de reglas y comportamientos que son configurables para responder a una variedad de escenarios de fallo. Al ser una base de reglas, también es extensible a lo largo del tiempo. Simplemente agregando reglas se puede extender la capacidad del sistema para manejar fallos

nuevos, emplear nuevos comportamientos o tomar decisiones diferentes en función de la información disponible. El controlador de fallos consultará la Base de Conocimientos utilizando la información disponible sobre el fallo actual. En el ejemplo actual, el controlador de fallos indicaría a la Base de Conocimientos que se produjo un evento malloc_failed dentro del programa actual. La Base de Conocimientos invocaría su base de reglas para tomar una decisión basada en reglas sobre la aplicación y el estado actual de la máquina, el tipo de fallo que se ha producido, cualquier información de gravedad o de fallo extendido disponible con el fallo o cualquier reglas de manejo de fallo específico de la aplicación. La Base de Conocimientos se puede implementar de varias maneras, incluidos los árboles de decisión, los razonadores basados en casos o las combinaciones de técnicas. El resultado del procesamiento de la Base de Conocimientos es la selección de una acción, o de ninguna acción, para tomar en respuesta al fallo. Esta instrucción de retorno indicará al controlador de fallos cómo proceder con el fallo actual.

El detector de fallos 201 también responde a eventos externos en el ordenador objetivo 101, o en la red de la que es parte. Por ejemplo, el ordenador objetivo 101 puede indicar que su potencia es baja, como cuando un dispositivo móvil funciona con baterías. El sistema de detección de fallos 201 puede observar este evento e invocar al controlador de fallos para seleccionar una acción de recuperación apropiada por medio de la Base de Conocimientos. En otras implementaciones, el detector de fallos 201 puede estar configurado para responder a pérdida de conectividad de red, fallo de servicios externos requeridos por la aplicación, fallo de dispositivos conectados tales como impresoras y módems, fallo de ordenadores de pares o servidores en su red, y otros eventos importantes. Se puede ver que el detector de fallos contiene dos módulos diferentes, uno para observar fallos específicos de la aplicación y otro para observar fallos generales del sistema o problemas ambientales.

El detector de fallos 201 se puede usar en conexión con un dispositivo móvil, capaz de desconectarse de una red. En este tipo de dispositivo, las interrupciones de la red pueden ser frecuentes y el código de la aplicación primaria 100 puede no estar diseñado para la movilidad. Además, en dispositivos móviles, el código de la aplicación primaria 100 puede recibir transmisión en continuo, un modo de entrega en el que la aplicación primaria 100 es dividida en segmentos de código que son necesarios para operar ciertas funciones de la aplicación primaria 100. Estos segmentos de código son entregados al dispositivo móvil justo a tiempo, de manera que la aplicación primaria 100 se pueda desplegar rápidamente, con una huella mínima. En este modo, es común que el operador del dispositivo móvil solicite una función de la aplicación primaria 100 durante el uso fuera de línea que puede no estar disponible. En el momento en que se solicita esta función, el segmento de código no estará presente y no será accesible desde la red.

En un ejemplo de esta invención, si el sistema operativo del dispositivo móvil emite un fallo de página que indica un fallo al acceder a una página de código del sistema de la aplicación primaria, el detector de fallos 201 atrapa este fallo e invoca al controlador de fallos 202. En un ejemplo alternativo, el detector de fallos 201 está insertado dentro del sistema operativo del dispositivo móvil, como parte del sistema para transmitir la aplicación. En este sistema, el fallo puede ser detectado antes de que se produzca el fallo de página, cuando la aplicación de red para el paquete de datos falla y el comportamiento de fallo es detectado directamente.

Puesto que el dispositivo móvil puede desconectarse de la red, puede perder contacto o puede ser incapaz de contactar con los recursos de la red. Si la aplicación primaria 100 intenta y no puede conectarse a la red, puede ocurrir otra clase de fallos. Con esta clase de fallos, el detector de fallos 201 puede interceptar la solicitud a la red y ejecutar un comportamiento de fallo apropiado en caso de fallo.

Como se ha hecho notar más arriba, después de que se detecta un fallo, el controlador de fallos 202 determina qué controlador de fallos es el más apropiado. En algunos casos, el controlador de fallos invocará la aplicación de asistencia 102 para permitir al usuario decidir sobre un curso de acción, o invocar acciones que puedan ser configuradas por la aplicación de asistencia. La aplicación de asistencia puede notificar al usuario de la presencia de un fallo, registrar el fallo internamente o reenviar la notificación de fallo a varios sistemas de monitorizado tales como SNMP, WMI o el Registro de Eventos de Windows. La aplicación de asistencia también puede preguntar al usuario si elige manejar el fallo o permitir que ocurra y, además, le pide al usuario que elija una acción apropiada si muchas son posibles.

Los controladores de fallos primarios incluyen un controlador de redirección, un controlador de suspensiones y un controlador de instantáneas. Cada uno de estos controladores de fallos a continuación se describirá en detalle en lo que sigue.

Controlador de redirección

Cuando se invoca el controlador de redirección, el controlador de redirección redirige la aplicación primaria a la lógica de "salvaguardar y salir" de la aplicación primaria y a continuación sale. En la figura 3 se muestra una realización de la invención que utiliza el controlador de redirección. En el punto en el que se ha producido el fallo, la aplicación primaria 100 se está ejecutando dentro de la rutina de interceptación del detector de fallos 201. El contador de programa del sistema operativo apuntará actualmente a esta rutina de interceptación y normalmente continuaría ejecutando la siguiente instrucción después de la llamada a la API defectuosa. El controlador de redirección invocará alternativamente la lógica de salvaguardar 302 de la aplicación y a continuación saldrá de la aplicación primaria 100.

En un ejemplo, esta invocación de la lógica de salvaguardar se implementa mediante (a) llamando directamente al punto de entrada de salvaguarda de la aplicación primaria 100, (b) restableciendo el contador de programa al punto de entrada de salvaguarda, o (c) insertando una instrucción de salto en el programa de llamada para forzar la llamada de la lógica de salvaguardar y salir.

En el ejemplo que se muestra en la figura 3, la aplicación primaria intenta llamar a la API call_thesaurus (). El segmento de código para esta rutina no ha sido paginado por el sistema operativo 303. Por lo tanto, los sistemas operativos 303 interceptan la llamada y consultan a su administrador de memoria virtual 304 para recuperar la página de códigos. Cuando esto falla, el sistema operativo 303 generará un fallo de página, normalmente terminando la aplicación primaria 100. El controlador de redirección 306 intercepta entonces el fallo de página. Alternativamente, el controlador de redirección 306 puede recibir la página "fallo en la recuperación del código" directamente de la rutina.

En algunos casos, la aplicación primaria puede tener varias variantes para archivar y salir. Puede haber "salvaguardar como", "desconectar" o "salir" que encapsula la funcionalidad de salvaguarda. El controlador de fallos 306 asegura que se le proporcione al usuario un medio seguro para salvaguardar el trabajo que se ha realizado y salir de la aplicación, evitando un fallo y / o una colisión de la aplicación. El usuario puede reiniciar la aplicación después de salir, pero sabrá que el fallo ha ocurrido y puede ocurrir nuevamente.

En una realización del controlador de fallos, la aplicación primaria invoca su lógica de "salvaguardar", pero su archivo de datos será redirigido a un origen de datos local, o almacén proxy, para que la aplicación primaria pueda completar su operación de salvaguardar en el caso en el que el archivo de datos está corrompido y provoque fallos de escritura, o es un archivo de red y la red no está disponible. Normalmente, el programa generaría un fallo adicional ya que no podría comunicarse con el servidor de archivos de red o salvaguardar su archivo de manera efectiva. Al reconectarse, el archivo de datos o el almacén proxy pueden sincronizarse o reemplazarse con el archivo de datos de origen.

Controlador de suspensiones

Cuando se invoca el controlador de suspensiones, el controlador de suspensiones suspende todos los procesos, elementos secundarios y subprocesos que comprenden la aplicación primaria 100. Una vez que se ha suspendido la aplicación primaria 100, el controlador de suspensiones espera la notificación de que se puede reanudar (por ejemplo, un evento tal como la red se restablece o el usuario selecciona un control para indicar su deseo de continuar). En la solicitud para continuar, el controlador de suspensiones puede (a) reanudar la aplicación primaria directamente, (b) restablecer el contador del programa, o (c) reinvocar la API originalmente defectuosa.

Durante la suspensión, la aplicación primaria 100 seguirá existiendo en su forma completa y seguirá siendo visible como una aplicación en el ordenador objetivo 101. En una realización, el controlador de suspensiones evitará la reanudación de la dispersión de la aplicación primaria por medio de herramientas de sistema simples tales como el Administrador de Tareas de Windows. Se suspenderá la aplicación primaria para que sea controlable solo a través de medios explícitos o a través de la aplicación de asistencia interna 102.

El controlador de suspensiones se usa a menudo para aplicaciones en las que no hay una funcionalidad explícita de "salvaguardar". La aplicación salvaguarda su estado como una serie de efectos laterales, o mediante la comunicación con un servidor externo, una base de datos u otro sistema. En un ejemplo, la aplicación es una aplicación de hipertexto y se pierde la conexión con el servidor HTTP. En este ejemplo, el estado lateral del usuario de la aplicación de hipertexto puede ser almacenado o la aplicación de navegador de hipertexto completa puede ser almacenada. Cuando el servidor vuelve a estar en línea, la aplicación puede continuar, utilizando su estado salvaguardado.

Controlador de instantáneas

Si el fallo indica un problema a largo plazo, y la suspensión no sobrevivirá a la duración del problema, puesto que el ordenador objetivo puede realizar un ciclo de conexiones, o sufrir fallos adicionales, y se invoca al controlador de instantáneas. En esta situación, se desea hacer una copia de seguridad del estado de ejecución de la aplicación primaria, para una restauración posterior.

Cuando se invoca el controlador de instantáneas, el controlador de instantáneas suspende los procesos y subprocesos de la aplicación primaria, toma una instantánea de la memoria de la aplicación primaria, incluidas las estructuras de datos del núcleo que no forman parte de la aplicación pero que se utilizan para restaurarla como un proceso. Esta instantánea de memoria se puede escribir en el disco para una recuperación posterior. En un procedimiento alternativo, el controlador de instantáneas puede hacer una captura proactiva de forma periódica de la aplicación primaria para proporcionar un medio de recuperación de grano más fino. Para evitar una sobrecarga abrumadora, la instantánea de memoria se puede tomar rápidamente, y almacenar en segundo plano la memoria en el disco u otro almacenamiento no volátil. Una vez que se completa la instantánea, el controlador de instantáneas puede finalizar la aplicación primaria si se está tomando la instantánea debido a un problema a largo plazo.

Una vez que se ha terminado la aplicación primaria, la aplicación de asistencia 102 puede entonces actuar para evitar que se vuelva a invocar la aplicación hasta que se haya resuelto el fallo, o el usuario haya confirmado explícitamente que el usuario desea que se vuelva a invocar la aplicación. En la invocación, la aplicación de asistencia puede restaurar la aplicación primaria 100 o simplemente permitir que por el contrario se cree otra instancia de la aplicación primaria. Alternativamente, la aplicación de asistencia puede transferir la imagen de instantánea a un ordenador de par o servidor en la red para que se reanude allí.

En la restauración, el controlador de fallos correlacionará la memoria del proceso y restablecerá las estructuras internas de datos del núcleo del sistema operativo para garantizar que el programa pueda continuar exactamente como se había dejado. Como ejemplo, si la aplicación 100 tiene varios archivos abiertos en el momento del fallo, el controlador de instantáneas deberá volver a abrir los archivos y reasignar los controladores de archivos conocidos para que la aplicación primaria no genere fallos al intentar usar sus antiguos controladores de archivos. Además, los controladores de archivos deben señalar el desplazamiento apropiado en el archivo si están configurados para el acceso a la transmisión en continuo.

La figura 5 muestra un diagrama de flujo de una realización de la aplicación de asistencia de la presente invención. En las etapas 501, 502 se llama a una API potencialmente defectuosa y la aplicación de asistencia 102 pregunta si la llamada fue exitosa o si se ha producido un fallo en la etapa 504. Si no se ha producido un fallo, la aplicación devuelve los resultados de API en la etapa 505. Si se ha producido un fallo, la aplicación de asistencia 102 identifica el controlador de fallos, eligiendo entre el controlador de redirección y el controlador de suspensiones. Si el controlador de suspensiones se selecciona en la etapa 508, el programa auxiliar pregunta si la aplicación primaria puede ser reanudada. Si se reanuda el programa, devuelve los resultados de la API en la etapa 505. Si la aplicación primaria no se reanuda, la aplicación de asistencia pregunta si debe reinvocar la API en la etapa 510. Si no se debe reinvocar la API, la aplicación de asistencia pregunta si una instantánea está disponible. Si hay una instantánea disponible, en la etapa 511, la aplicación de asistencia revierte el PC a la instantánea en la etapa 513. Si la instantánea no está disponible, restablece el contador del programa en la etapa 512.

En todavía otro ejemplo de esta invención, la aplicación de asistencia 102 también restaurará conexiones externas tales como a una base de datos, un zócalo TCP / IP u otro mecanismo IPC. Para hacerlo, la aplicación de asistencia 102 puede pedir ayuda al usuario, tal como con el inicio de sesión de la base de datos o la autenticación con un servidor HTTP. Como se ha descrito más arriba, el sistema puede necesitar reiniciar algunos elementos internos de la aplicación primaria o redirigir los identificadores a entidades que han cambiado, como un descriptor de zócalo. En una realización alternativa, se puede proporcionar una interfaz de usuario para permitir al usuario invocar explícitamente este comportamiento, independientemente de la existencia o inexistencia de fallos.

La aplicación de asistencia clasifica qué conjunto de fallos manejará además de qué controladores puede invocar. La aplicación de asistencia puede obtener este conocimiento fuera de línea o en línea por medio de una variedad de técnicas. En un ejemplo, el controlador de fallos registrará para detectar todos los fallos.

En otra realización de la invención, se toma una etapa adicional para realizar análisis estático en el código de aplicación primaria para determinar su composición y potencial para fallos, así como sus puntos de entrada de programa comunes tales como "salvaguardar" o "salir". En el examen de fallos potenciales, las rutinas de análisis buscan el uso de un conjunto de API comunes como acceso a archivos o bases de datos por medio de ODBC, o como se ha descrito más arriba el uso de rutinas de memoria como malloc () y free () o rutinas IPC tales como zócalo () o Windows GetNamedPipe ().

Una vez que se completa el análisis, el analizador 403 puede presentar una lista de los fallos que se sabe que son recuperables por medio de los controladores de fallos disponibles en el sistema y los que no son recuperables y las recomendaciones para manejar los fallos (véase la figura 4). Para ambos conjuntos de fallos, las acciones pueden ser proscritas para configurar los controladores de fallos. Para aquellos fallos que no son recuperables, el controlador de redirección se usa comúnmente. Si no existe un código de "salvaguardar" o es identificable, el sistema puede configurarse para simplemente salir fácilmente.

Para analizar los puntos de entrada del programa, el sistema puede usar el conocimiento de la plataforma para descubrir las fuentes. Por ejemplo, en la plataforma Microsoft Windows, un controlador de mensajes que puede responder al mensaje WM_EXIT puede corresponder a la rutina de "salida" del programa. Por lo tanto, una búsqueda de WM_EXIT indicará este punto de función. Alternativamente, un archivo de recursos puede describir el comando de menú Archivo: Salir y el mensaje de Windows que generará en la invocación. Si estos indicadores no existen, el analizador puede ir más allá para buscar el uso de API como las funciones Windows ExitProcess () o exit (). Al buscar la rutina de "salvaguardar", el analizador puede buscar cualquier uso de archivos o fuentes de datos externas.

En una realización alternativa, los fallos y los puntos de entrada se pueden identificar usando análisis dinámico. El programa puede ser rastreado durante su ejecución para identificar estas firmas. En una realización, el análisis dinámico se realiza con un identificador de punto de entrada UI 404 que registra los puntos de entrada de una manera similar a la macrograbación. Puesto que el usuario indica que identificará la función "salvaguardar", el identificador

registra toda la UI y la actividad del programa. A partir de estas firmas, se puede identificar el punto de entrada y el código asociado. Para las aplicaciones de transmisión en las que la aplicación se usa sin conexión, el código identificado con las funciones "salvaguardar" y "salir" se etiqueta especialmente para indicar que siempre se debe transmitir y presentar en el modo fuera de línea para garantizar que se pueda invocar este comportamiento.

La presente invención también proporciona un medio para permitir la interrupción en el uso o de fallos en una aplicación que debe ser acomodada sin la pérdida de trabajo. Normalmente, el fallo durante el uso de una aplicación provocaría que se abandone una cierta cantidad de trabajo realizado en ese programa, ya sea completo o no. El sistema y el procedimiento de la presente invención proporcionan más que una simple recuperación del programa, sino que por el contrario permiten un medio para asegurar que no se pierda ningún dato. Además, se realiza sin requerir ninguna modificación, recopilación o rediseño de software, y permite que se cree un nuevo software con este paradigma en mente.

Aunque la invención se ha descrito en conexión con ciertas realizaciones preferidas, se entenderá que no se pretende limitar la invención a esas realizaciones particulares. Por el contrario, está destinada a cubrir también alternativas, modificaciones y equivalentes. Se mencionan algunos componentes, figuras y tipos de materiales específicos, pero se debe entender que tales valores de componentes, dimensiones y tipos de materiales se proporcionan, sin embargo, como ejemplos solamente y no están destinados a limitar el alcance de esta invención de ninguna manera..

REIVINDICACIONES

1. Un procedimiento para minimizar la pérdida de datos durante el uso de una aplicación informática (100, 200, 300), que comprende las etapas de:

   monitorizar una llamada desde la aplicación informática a una rutina para detectar (504) la presencia de un fallo, en el que la etapa de monitorizar la llamada desde la aplicación informática a la rutina comprende monitorizar la salida de la llamada rutinaria para fallos tales como excepciones, malos códigos de retorno u otro comportamiento incorrecto, y al regresar, inspeccionar los resultados de la llamada;

   si no se ha producido un fallo, devolver el control a la aplicación informática; y

   si se ha producido un fallo,

   seleccionar (506) un procedimiento para manejar el fallo, seleccionándose el procedimiento para manejar el fallo de los siguientes procedimientos: redirigir (507) la ejecución del código de la aplicación informática y suspender (508) la aplicación informática; y realizar el procedimiento seleccionado para manejar el fallo, en el que realizar una redirección comprende invocar una lógica de salvaguarda identificada de la aplicación informática para salvaguardar datos.

2. El procedimiento de la reivindicación 1, en el que realizar el procedimiento seleccionado para manejar el fallo comprende comunicar (204) con un usuario de la aplicación informática con respecto al fallo.

3. El procedimiento de la reivindicación 1, que comprende, además, registrar el fallo internamente en el sistema que ejecuta la aplicación informática.

4. El procedimiento de la reivindicación 1 que comprende, además, reenviar una notificación de fallo a un sistema de supervisión externo.

5. El procedimiento de la reivindicación 1 que comprende, además, comunicar con un usuario de la aplicación informática con respecto al fallo.

6. El procedimiento de la reivindicación 5, en el que la etapa de comunicar con un usuario de la aplicación informática con respecto al fallo comprende preguntar al usuario (204) si quiere que se maneje el fallo o si se debe permitir que se produzca el fallo.

7. El procedimiento de la reivindicación 1, en el que cuando el procedimiento seleccionado es suspender (508) la aplicación informática en respuesta a la detección de un fallo, el procedimiento comprende además:

   monitorizar (509) la aplicación informática para indicar que se puede producir la reanudación; y

   seleccionar uno de entre reanudar la aplicación informática o restablecer (512) a un contador de programa de la aplicación informática.

8. El procedimiento de la reivindicación 7, que comprende, además, evitar la reanudación no deseada de la aplicación informática mientras la aplicación informática está suspendida.

9. El procedimiento de la reivindicación 8, en el que la etapa de evitar la reanudación no deseada de la aplicación informática comprende hacer que la aplicación informática se pueda controlar solamente con medios predeterminados.

10. El procedimiento de la reivindicación 7, en el que la aplicación informática es suspendida durante la duración de un fallo detectado.

11. El procedimiento de la reivindicación 7, en el que la etapa de suspender la aplicación informática comprende la etapa de comunicar con un usuario por medio de una interfaz de usuario para determinar si el usuario desea tener la aplicación informática suspendida en anticipación a un estado de fallo.

12. El procedimiento de la reivindicación 1, en el que cuando el procedimiento seleccionado suspende (508) la aplicación informática, el procedimiento comprende además almacenar una instantánea de memoria en un dispositivo de memoria auxiliar.

13. El procedimiento de la reivindicación 12 que comprende, además :

   terminar la aplicación informática después de que se almacene la instantánea de memoria; y

   en la invocación, restaurar la aplicación informática o permitir que se cree otra instancia de la aplicación informática.

14. El procedimiento de la reivindicación 13, en el que la instantánea de memoria almacenada es recuperable después de la terminación de la aplicación informática.

15. El procedimiento de la reivindicación 12, que comprende, además: evitar que se invoque la aplicación informática después de la suspensión de la aplicación informática hasta que un usuario elimine o confirme explícitamente un fallo detectado.

16. El procedimiento de la reivindicación 13, en el que la etapa de restaurar la aplicación informática comprende mapear la memoria de la aplicación informática y restablecer las estructuras internas de datos del núcleo del sistema operativo.

17. El procedimiento de la reivindicación 12, que comprende, además, las etapas de restaurar una conexión externa a un ordenador en el que se usa la aplicación informática en respuesta a la detección de un fallo de que la conexión externa está desconectada.

18. El procedimiento de la reivindicación 1, que comprende, además, analizar (403) el código (402) de la aplicación informática para identificar los puntos de entrada de salvaguarda.

19. El procedimiento de la reivindicación 18, en el que la etapa de analizar el código para identificar los puntos de entrada de salvaguarda comprende rastrear la aplicación durante su ejecución.

20. Un sistema informático dispuesto para realizar el procedimiento de una de las reivindicaciones 1 a 19.

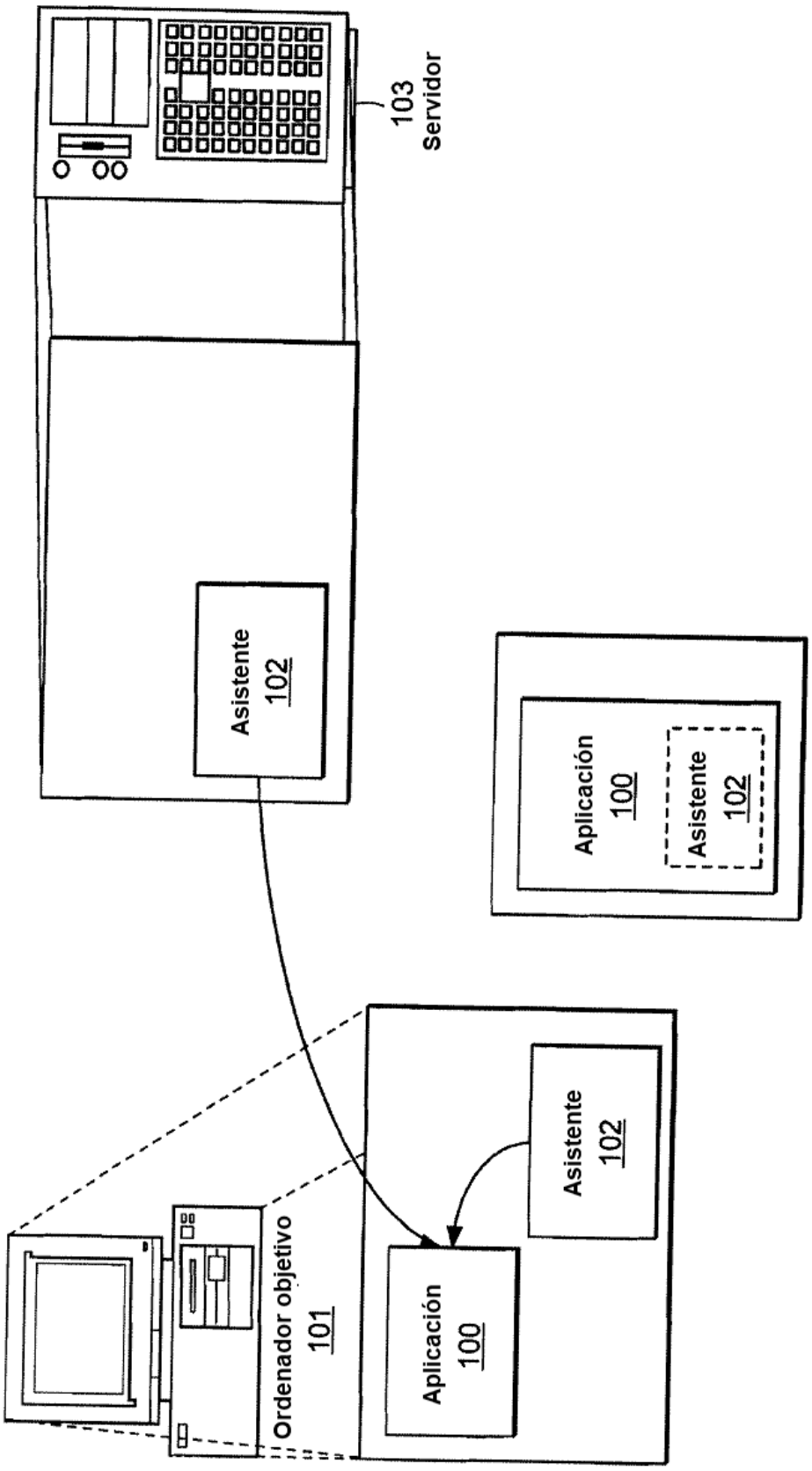
103
Servidor
Asistente
102
Aplicación
100
Asistente
102
Asistente
102
Aplicación
100
Ordenador objetivo
101

FIG. 1

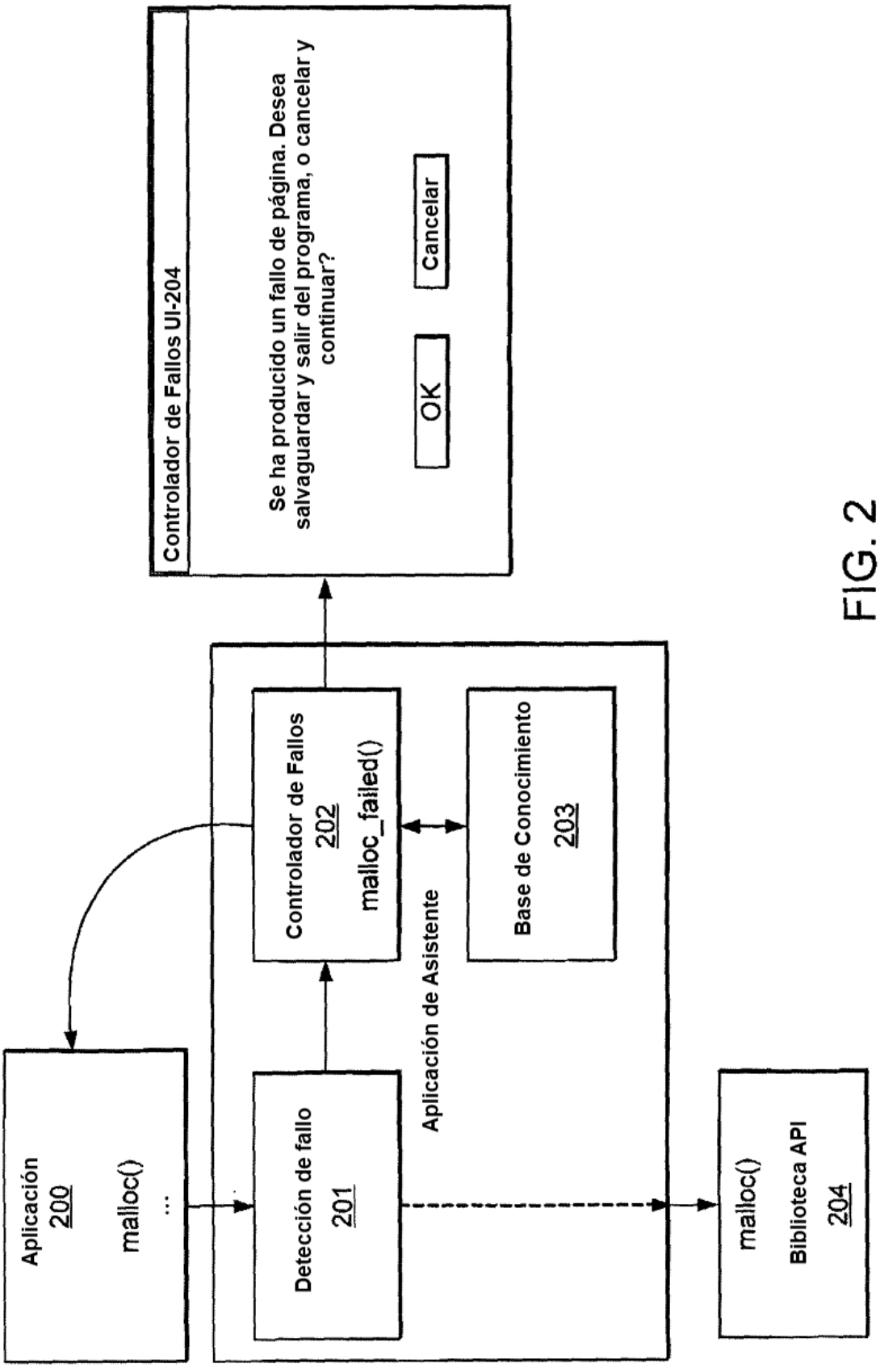
Controlador de Fallos UI-204
Se ha producido un fallo de página. Desea salvaguardar y salir del programa, o cancelar y continuar?
OK
Cancelar
Aplicación
200
malloc()
...
Detección de fallo
201
Controlador de Fallos
202
malloc_failed()
Base de Conocimiento
203
Aplicación de Asistente
malloc()
Biblioteca API
204

FIG. 2

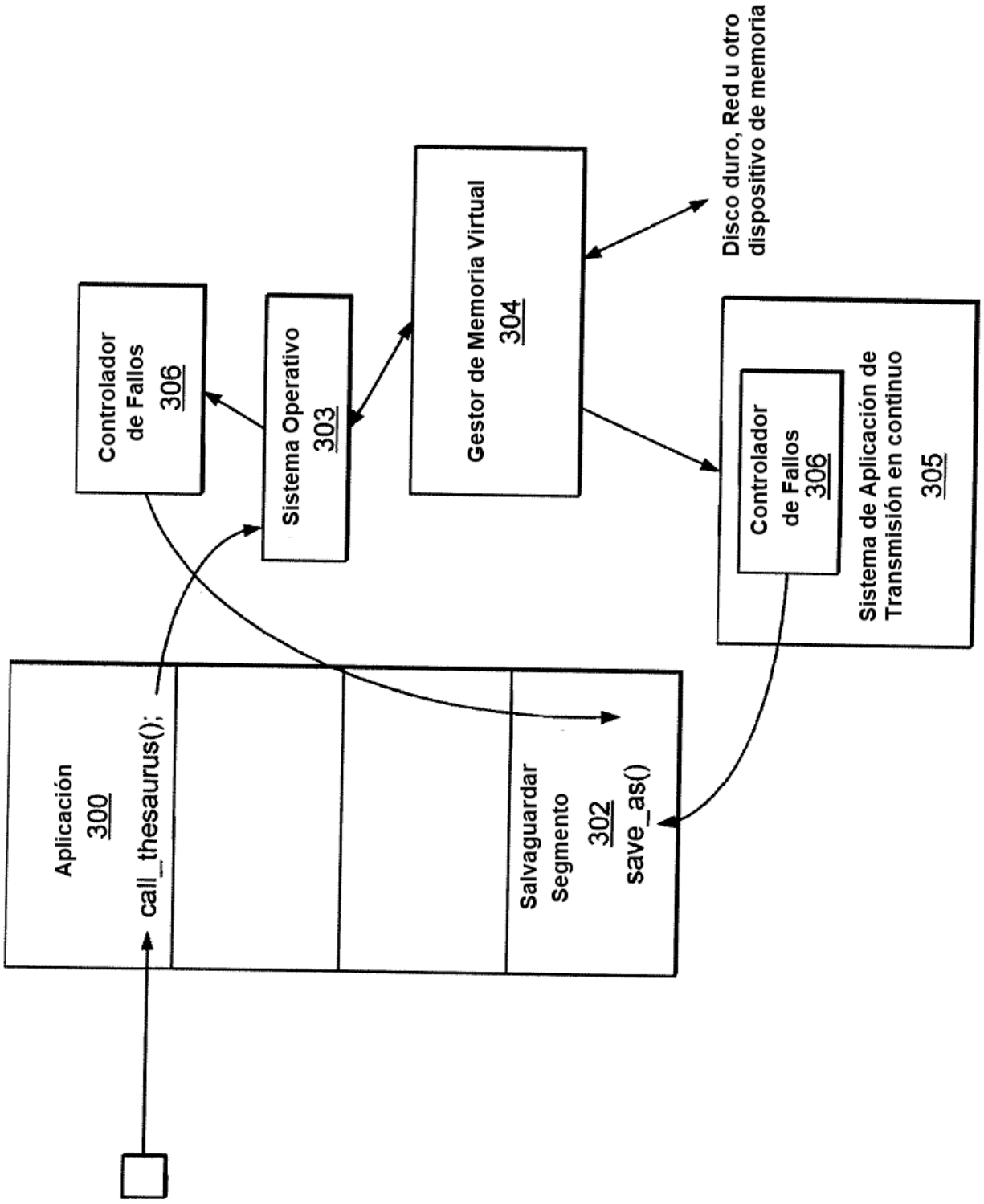
Aplicación
300
call_thesaurus();
Salvaguardar
Segmento
302
save_as()
Controlador
de Fallos
306
Sistema Operativo
303
Gestor de Memoria Virtual
304
Disco duro, Red u otro
dispositivo de memoria
Controlador
de Fallos
306
Sistema de Aplicación de
Transmisión en continuo
305

FIG. 3

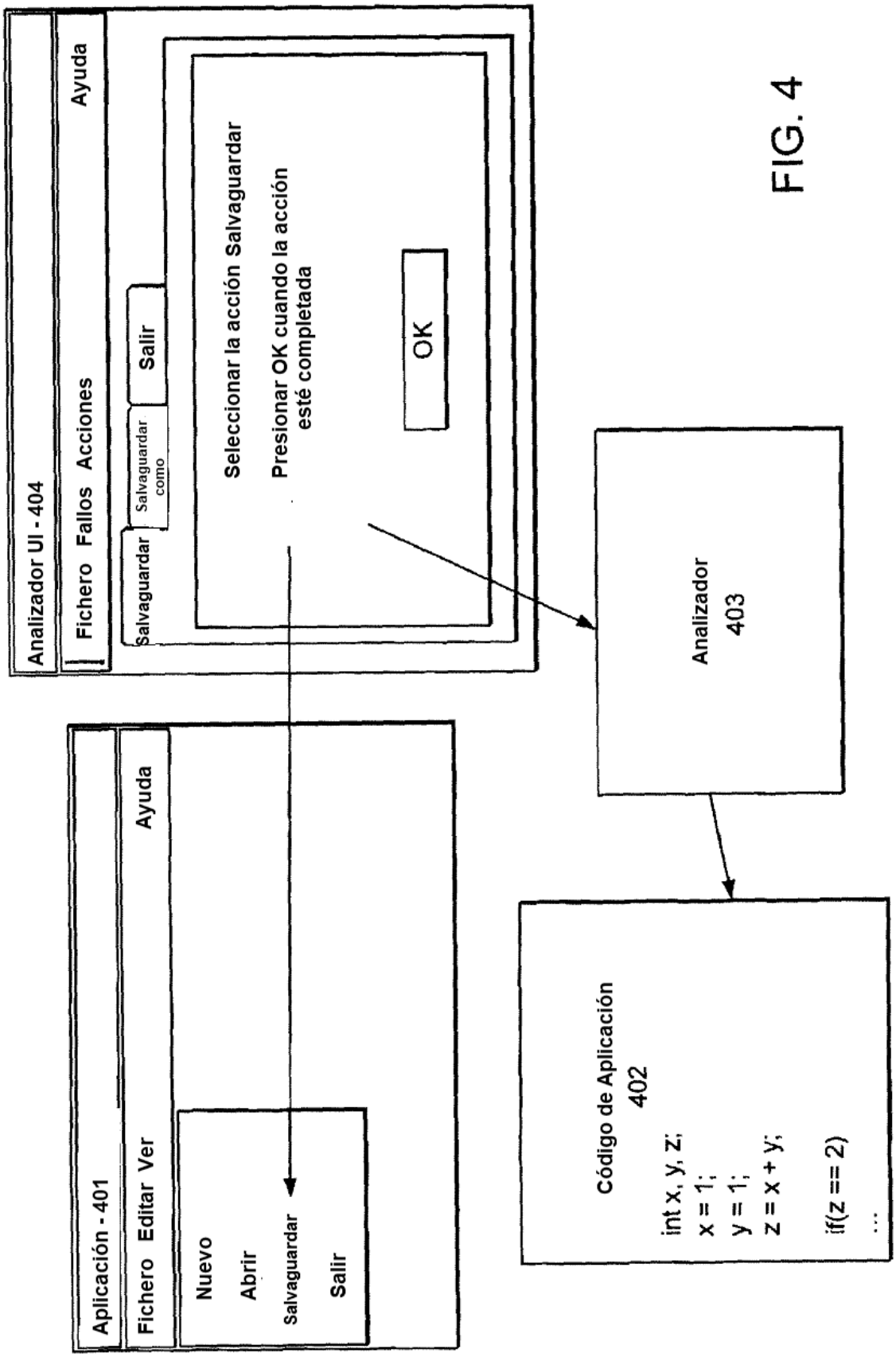
Analizador UI - 404
Fichero Fallos Acciones
Ayuda
Salvaguardar
Salvaguardar como
Salir
Seleccionar la acción Salvaguardar
Presionar OK cuando la acción esté completada
OK
Aplicación - 401
Fichero Editar Ver
Ayuda
Nuevo
Abrir
Salvaguardar
Salir
Analizador
403
Código de Aplicación
402
int x, y, z;
x = 1;
y = 1;
z = x + y;
if(z == 2)
...

FIG. 4

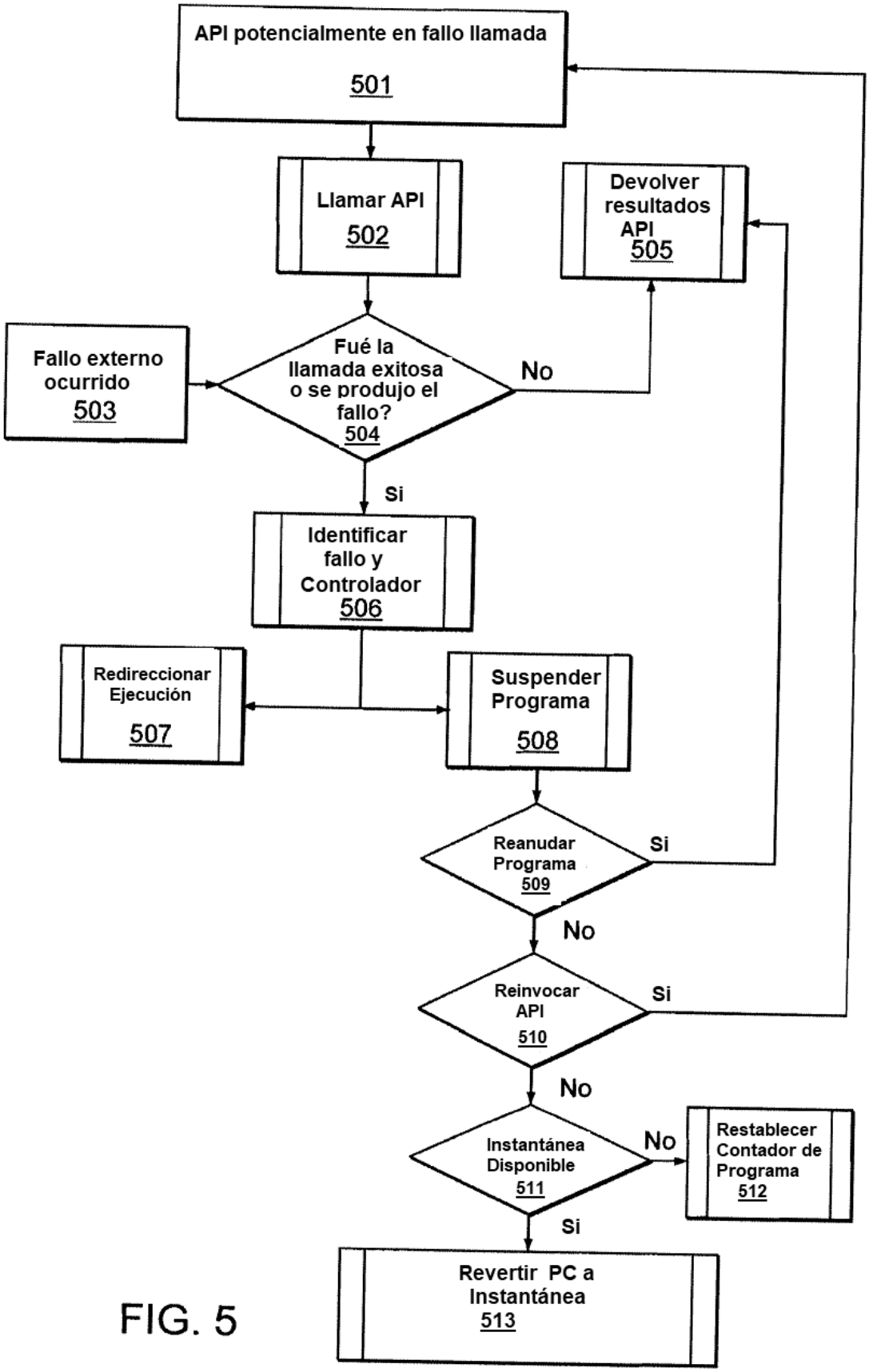
API potencialmente en fallo llamada
501
Llamar API
502
Devolver resultados API
505
Fallo externo ocurrido
503
Fué la llamada exitosa o se produjo el fallo?
504
No
Si
Identificar fallo y Controlador
506
Redireccionar Ejecución
507
Suspender Programa
508
Reanudar Programa
509
Si
No
Reinvocar API
510
Si
No
Instantánea Disponible
511
No
Restablecer Contador de Programa
512
Si
Revertir PC a Instantánea
513

FIG. 5