Industria y Comercio

SUPERINTENDENCIA

SOLICITUD
PATENTE DE INVENCIÓN

EXPEDIENTE N° 05-32905

TITULO: PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE

CLASIFICACION INTERNACIONAL: G06F 17/60

SOLICITANTE: MICROSOFT CORPORATION

DOMICILIO: REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA

APODERADO: DARIO CARDENAS NAVAS

BOGOTÁ, D.C. _____

020414

PETITORIO

SUPERINTENDENCIA Y COMERCIO
Radicacion : 05042905 00000000 Folios: 270
Fecha (AMD): 2005-04-11 16:10:14
Tramite : 002 PATENTES 0 1 REGISTRO/ 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS SOLICITUDES

Industria y Comercio
SUPERINTENDENCIA

FORMULARIO UNICO DE SOLICITUD DE PATENTE
2000-3

1 **SOLICITUD DE:**

[X] Patente de Invención [ ] Patente de Modelo de Utilidad

SOLICITANTE (71)	Nombre:	**MICROSOFT CORPORATION**	**IDENTIFICACION**
	Dirección:	One Microsoft Way Redmond, Washington 98052, Estados Unidos de America	C.C. [ ] NIT [ ]
	Teléfono:	3123600 Fax: 3122410	C.E. [ ] Otro [X]
	E-Mail:	general@cardenasycardenas.com	
	Lugar de Constitución	Estado de Washington, Estados Unidos de América	Número
	Domicilio:	REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA.	
REPRESENTANTE O APODERADO (74)	Nombre:	**DARIO CARDENAS NAVAS**	**IDENTIFICACION**
	Dirección:	Carrera 7 No 71-52 Torre B Piso 9	C.C. [X] NIT [ ]
	Teléfono:	3123600 Fax: 3122410	C.E. [ ] Otro [ ]
	E-Mail:	general@cardenasycardenas.com	Número 17.066.629 BTA TP 1.250 C.S.J.
INVENTOR(ES) (72)	Nombre:	1. Anthony Blumfield 2. Gilad Golan 3. Jason Garms 4. Saud A. Alshibani 5. Scott A. Field	**IDENTIFICACION**
	Dirección:	1. Gran Bretaña 2. Israel 3. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America 4. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America 5. Gran Bretaña	C.C. [ ] NIT [ ]
	Teléfono:	3123600 Fax: 3123600	C.E. [ ] Otro [X]
	E-Mail:	general@cardenasycardenas.com	
	Domicilio:	1. GRAN BRETAÑA 2. ISRAEL 3. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA. 4. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA. 5. GRAN BRETAÑA	

5 **Título(54):**

PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE

6 **Prioridad** SI [X] NO [ ]

(33)País de Origen	(32)Fecha	(31)N° de Solicitud
USA	Mayo 11, 2004	60/570,124
USA	Junio 30, 2004	10/880,709

Para publicar a partir de la fecha de la presente solicitud a los:

[ ] 6 meses [ ] 12 meses [ ] 18 meses [ ] Otro

CLASIFICACION INTERNACIONAL
G06F 17/60

Comprobante de Pago No.: 05-23,583/05-23,584/05-23,585 Abril 5, 2005

ANEXOS

2

- [X] Comprobante de pago de la tasa de presentación de la solicitud. (No.05-23,583)
- [ ] Comprobante de pago de la tasa por concepto de excedente de palabras en la publicación.
- [X] Comprobante de pago de la tasa por concepto de reivindicación de prioridad. (No.05-23,584/05-23,585)
- [X] Poderes, si fuere el caso. (Protocolo No 16.069)
- [X] Certificado de existencia y representación legal cuando el solicitante sea persona jurídica. Protocolo No 16.069)
- [X] Copia certificada de la primera solicitud, si se reivindica prioridad.
- [X] Traducción simple de la primera solicitud, si se reivindica prioridad.
- [X] Documento de cesión del inventor al solicitante o a su causante.
- [X] Descripción de la invención.
- [X] Una o más reivindicaciones.
- [X] Dibujos y/o planos necesarios.
- [ ] De ser el caso, copia del contrato de acceso.
- [ ] De ser el caso, documento que acredite la licencia o autorización de uso de conocimientos tradicionales de la comunidad de las comunidades indígenas.
- [ ] De ser el caso, certificado de depósito del material biológico.
- [X] Arte final 12x12 y 6x6 cm.
- [X] Resumen y extracto para publicación.

NOMBRE: DARIO CARDENAS NAVAS

FIRMA:

C.C. 17.066.629 BTA TP. 1.250 C.S.J.

SUPERINDUSTRIA Y COMERCIO
Radicacion : 05032905 00000000 Folios: 270
Fecha (AMD): 2005-04-11 16:10:14
Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO

NIT : 800176089-2

RECIBO OFICIAL DE CAJA : 05 - 23.583
FECHA : ABRIL 5 DE 2005

***** C O N S I G N A C I O N *****

DEPOSITANTE	TIPO PAGO	BANCO	CUENTA	No. PAGO	FECHA PGO	Vr. PAGO
CARDENAS Y CARDENAS P.I. CONSIGNACION		BANCO POPULAR	050-00110-6	32634199	05/04/2005	8,118.000.00

***** C O N C E P T O *****

CANT.	RENTISTICO	CONCEPTO	TOTAL CONCEPTO
1	50005-01-01 SOLICITUDES	1 TRAMITES DE SOL. DE PATENTE DE IN	443,000.00
		TOTAL : $	443.000.

SON: CUATROCIENTOS CUARENTA Y TRES MIL PESOS

RESPONSABLE : _____

RECIBO DE CAJA APLICADO AL EXPEDIENTE No. _____

PARCHEO (CONEXION PROVISIONAL) EFICIENTE

CARDENAS & CARDENAS
ABOGADOS

4

SUPERINDUSTRIA Y COMERCIO

Radicacion : 05022905 00000000 Folios: 270

Fecha (AMD): 2005-04-11 16:10:14

Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTAC

Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO

NIT : 800176089-2

RECIBO OFICIAL DE CAJA : 05 - 23,584
FECHA : ABRIL 5 DE 2005

***** C O N S I G N A C I O N *****

DEPOSITANTE	TIPO PAGO	BANCO	CUENTA	No. PAGO	FECHA PGO	Vr. PAGO
CARDENAS Y CARDENAS P.I. CONSIGNACION		BANCO POPULAR	050-00110-6	32654199	05/04/2005	8.118.000.00

***** C O N C E P T O *****

CANT.	RENTISTICO	CONCEPTO	TOTAL CONCEPTO
1	50005-01-01 SOLICITUDES	82 INVOCACION DE UNA PRIORIDAD EN NU	124,000.00
		TOTAL : $	124,000.00

SON: CIENTO VEINTICUATRO MIL PESOS

RESPONSABLE : _____

RECIBO DE CAJA APLICADO AL EXPEDIENTE No._____

PARCHEO (CONEXION PROVISIONAL) EFICIENTE

CARDENAS & CARDENAS ABOGADOS

4

5

SUPERINDUSTRIA Y COMERCIO
Radicacion : 05032905 00000000 Folios: 270
Fecha (AMD): 2005-04-11 16:10:14
Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO

NIT : 800176089-2

RECIBO OFICIAL DE CAJA : 05 - 23.585
FECHA : ABRIL 5 DE 2005

C O N S I G N A C I O N

DEPOSITANTE	TIPO PAGO	BANCO	CUENTA	No. PAGO	FECHA PAG	Vr. PAGO
CARDENAS Y CARDENAS P.I. CONSIGNACION		BANCO POPULAR	050-00110-6	32654199	05/04/2005	8.118.000.00

C O N C E P T O

CANT.	RENTISTICO	CONCEPTO		TOTAL CONCEPTO
1	50005-01-01 SOLICITUDES	82	INVOCACION DE UNA PRIORIDAD EN NU	124,000.00
			TOTAL : $	124.000.00

SON: CIENTO VEINTICUATRO MIL PESOS

RESPONSABLE : _____

CARDENAS & CARDENAS

RECIBO DE CAJA APLICADO AL EXPEDIENTE No._____

PARCHEO (CONEXION PROVISIONAL) EFICIENTE

ABOGADOS

EL CAMPO TÉCNICO

[0001] La invención presente está dirigida al campo de actualización de la operación de programas de computadora instalados.

EL FONDO

[0002] "Reparchar" o remendar es el proceso de modificar los programas ya-instalados, incluso los programas de aplicación, programas utilitarios, sistemas operativos y componentes del sistema operativo, controladores de dispositivos, etc. Remendar puede ser útil para modificar los programas de una variedad de propósitos, incluyendo la corrección de un error de programación, reduciendo o eliminando un riesgo de seguridad, o mejorando la lógica usada por el programa modificado. El "Reparcheo" es iniciado típicamente por la compañía u otra organización que originalmente proporcionó el programa a ser remendado.

[0003] Los programas instalados están predominantemente compuestos de módulos de código ejecutables. Como ejemplo, muchos programas diseñados para ejecutarse sobre el sistema operativo Windows XP de Microsoft Corp. de Redmond, Washington están predominantemente compuestos de módulos de código ejecutables llamados "DLLs." Un acercamiento convencional popular para remendar es identificar, entre los módulos de código ejecutables que constituyen el programa instalado a ser remendado, el módulo de código ejecutable que contiene el código del programa que uno desea modificar con un parche; creando una nueva versión del módulo de código ejecutable identificado en el cual se desea que modificación sea hecha; y distribuyendo la nueva versión del módulo de código ejecutable identificado, junto con un programa instalador, a usuarios que pueden desear aplicar el parche. Cada usuario determina entonces si desea aplicar el parche, y en ese caso, ejecuta el programa instalador que

reemplaza la versión original del módulo del código ejecutable identificado con la nueva versión del módulo del código ejecutable identificado.

[0004] Los acercamientos convencionales para remendar tienen varios desventajas significantes. Estas desventajas aumentan a menudo la carga asociada con recibir y aplicar los parches. En algunos casos, este aumento de carga tarda la aplicación de algunos parches por algunos usuarios, e incluso previene la aplicación de algunos parches por algunos usuarios. Tal retraso y prevención en la aplicación de parches pueden en algunos casos tener consecuencias negativas serias para los usuarios, sobre todo para los parches diseñados para reducir o eliminar un riesgo de seguridad.

[0005] Una desventaja de acercamientos convencionales a remendar relacionada a casos comunes en que los parches múltiples deben crearse distribuidos para efectuar una sola modificación a un solo programa. En algunos casos, el programa a ser remendado tiene varios "sabores" diferentes de un módulo de código ejecutable particular, como un sabor diferente para cada sistema operativo o versión del sistema operativo en que el programa esta diseñado para ejecutarse, y/o cada versión del idioma natural del programa. Donde el módulo de código ejecutable identificado es tal un módulo de código ejecutable, la creación del parche y el proceso de distribución descritos para cada sabor del módulo de código ejecutable identificado. El usuario debe seleccionar entonces y debe aplicar el parche para el sabor apropiado del módulo de código ejecutable identificado. El ordenamiento a través del número grande resultante de parches y seleccionar el conjunto apropiado de parches para la aplicación en el sistema de cómputo de cada usuario pueden ser muy pesados, tanto para que esta condición a veces se llame el "infierno del parche". En algunos casos, un administrador debe mantener una base de datos del inventario que identifica el conjunto de versiones del módulo ejecutable instalado en cada sistema designado que es usado para seleccionar los parches convencionales apropiados para cada sistema designado.

[0006] Otra desventaja de acercamientos convencionales para remendar se relaciona al tamaño grande de los parches distribuidos. No es raro para los módulos de código ejecutables tener un tamaño medido en megabytes que a su vez causa que un solo parche tenga ese tamaño comparable, haciéndolo pesado para distribuir y guardar, o incluso imposible de distribuir y guardar, para algunos usuarios. Este problema puede multiplicarse para parches que tienen múltiples sabores. Más allá, porque cada parche convencional incluye típicamente a un suplente completo del módulo ejecutable, mientras se aplican los parches convencionales pueden contribuir al problema de errores de código.

[0007] Una desventaja adicional al acercamiento convencional de remendar se relaciona a una necesidad para algunos usuarios de probar los parches antes de aplicarlos a los sistemas de cómputo de producción. En algunos casos, instalar un parche en un sistema de la computadora puede tener resultados adversos, como dónde la nueva versión del módulo de código ejecutable identificado contenida en el parche introduce un nuevo error de programación, o donde causa una nueva, interacción impredecible con otro programa que corre en el sistema de cómputo contra el cual es aplicado. En concordancia, a menudo antes de aplicar un parche contra un sistema de producción cuyos datos y funcionamiento son importantes de sostener, el usuario aplica el parche primero contra un sistema de prueba para evaluar si el parche es seguro de aplicar al sistema de producción. Tal comprobación separada de parches agrega carga asociada al "reparchamineto". Adicionalmente, dónde un parche convencional crea un problema--como un problema de compatibilidad de aplicación o una nueva vulnerabilidad – en un momento substancial después de que el parche es aplicado, puede ser difícil de rastrear tales problemas detrás del parche.

[0008] Una desventaja adicional de acercamientos convencionales para remendar se relaciona con el funcionamiento del instalador incluido en el parche. A menudo para reemplazar un módulo de código ejecutable que es parte de ejecutar el programa, el instalador debe terminar la ejecución de

ese programa primero. Tampoco, en algunos casos, tal reemplazo puede completarse sin reiniciar el sistema de cómputo. Estos dos pasos pueden causar rupturas sustanciales en el uso del sistema de cómputo remendado.

[0009] Otra desventaja de acercamientos convencionales a remendar involucra intentar remendar un módulo ejecutable para el cual "un parche privado", también llamado " parche caliente ó parche urgente – Hot Fix ", ha sido emitido antes para un subconjunto apropiado de clientes para ese módulo ejecutable. En tales casos, debido a dificultades encontradas en la distribución de parches convencionales que sustituyen nuevas versiones diferentes del módulo del código ejecutable que dependiendo de cada usuario que depende si el usuario ha aplicado el parche caliente, en cambio es típico distribuir un parche convencional simple que suple una sola nueva versión del módulo ejecutable independiente de si el usuario ha aplicado el parche caliente. Si esa nueva versión incluye el parche caliente, el parche impone el parche caliente en clientes no pensados para recibirlo. Si, por otro lado, esa nueva versión no incluye el parche caliente, priva a los clientes pensados para recibir el parche caliente del parche caliente.

[0010] Otra desventaja de acercamientos convencionales a remendar involucra el hecho de que los instaladores para productos de software que confían en un módulo ejecutable particular, como una biblioteca de enlace dinámica particular, a menudo "esconden" el módulo ejecutable guardándolo en una ubicación no-normal en el sistema de archivos (filesystem) del sistema de la computadora objetivo. En concordancia, a veces es difícil o imposible determinar si un sistema designado particular contiene una copia del módulo ejecutable a ser remendado, y, en ese caso, dónde reside en el sistema de archivos (filesystem) del sistema de cómputo designado. También, algunos productos de software mantienen un catálogo de versiones de módulo ejecutables que se han instalado por sus instaladores. Un producto de software puede contar con la exactitud de una indicación en el catálogo de versión de un módulo ejecutable particular. Tal confianza es derrotada donde un parche convencional reemplaza la versión

del módulo ejecutable identificada en el catálogo con una nueva versión del módulo ejecutable sin actualizar el catálogo.

[0011] Otra desventaja de acercamientos convencionales para remiendos sueltos desde el hecho que son imposibles de aplicar en un momento antes de que el módulo ejecutable a ser remendado se instala en el sistema de la computadora designada. Como resultado, si el módulo ejecutable a ser remendado se instala en el sistema de la computadora designada después de que un parche convencional para ese módulo ejecutable se recibe, es improbable que el parche se aplique al módulo ejecutable.

[0012] Otra desventaja de acercamientos convencionales para remendar es que ellos pueden aplicarse típicamente sólo por un usuario autenticado en el sistema de la computadora designada que usa una cuenta administrativa que tiene los permisos de modificación libres o habilitados. Autenticarse en una cuenta administrativa para este propósito puede hacer que el sistema de cómputo designado sea vulnerable a virus presentes en el sistema de la computadora designada que busca modificar aspectos del sistema de la computadora designada y permisos libres requeridos para eso.

[0013] Otra desventaja de acercamientos convencionales a remendar es que esos parches convencionales pueden ser difíciles o imposibles de desactivar, requiriendo pasos tales como reversar el reemplazo de un módulo ejecutable, o reversar una o más modificaciones al registro del sistema.

[0014] En concordancia, un nuevo acercamiento a remendar supera algunas o todas las desventajas de acercamientos convencionales para remendar lo discutido anteriormente tendría utilidad significante.

DESCRIPCIÓN BREVE DE LOS DIBUJOS

[0015] La figura 1 ilustra un ejemplo de un ambiente de sistema de informática conveniente en el cual la facilidad puede implementarse.

[0016] La figura 2 es un diagrama de flujo de datos que muestra un intercambio típico de datos entre los sistemas de la computadora de acuerdo con la facilidad.

[0017] La figura 3 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para recibir y procesar un nuevo parche.

[0018] La figura 4 es un diagrama de estructura de datos que muestra una tabla de parche de muestra típico de aquéllos usados por la facilidad.

[0019] La figura 5 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para poner al día las instrucciones de la configuración para un parche particular.

[0020] La figura 6 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para realizar la aprobación del parámetro especificado por un parche.

DESCRIPCIÓN DETALLADA

[0021] Una facilidad del software para remendar el código de programa de computadora instalado ("la facilidad") es proporcionada. En algunas personalizaciones, la facilidad agrega parámetros de prueba y prueban el resultado manejando las funciones instaladas. En otras personalizaciones, la facilidad agrega varios otros tipos de funcionalidades a las funciones instaladas, en algunos casos en posiciones arbitrarias en los flujos de ejecución de las funciones instaladas.

[0022] En algunas personalizaciones, para cada parche, la facilidad distribuye a cada sistema de computadora a ser remendado—es decir, cada "sistema de computadora" designado – la especificación de un punto en el cual realizar una prueba, la identidad de la prueba a realizar, y cómo actuar en respuesta a uno o más resultados de la prueba diferentes. En algunas personalizaciones, la facilidad proporciona un conjunto normal de validación de parámetro y otras pruebas cuyo uso puede especificarse en los parches. Por ejemplo, un parche puede especificar que, para una función particular,

si un parámetro particular de la función no tiene un cierto valor, la invocación de la función debe fallar antes de que su ejecución substantiva empiece. Otro parche puede especificar que, para una función particular, si un parámetro particular tiene una longitud que excede una longitud máxima especificada, el parámetro debe truncarse a la longitud máxima especificada antes de la ejecución de que la función se permita proceder. Muchos proezas de seguridad se basan en causar que funciones a ser llamadas con valores de parámetro que, mientras ellos no se bloqueen en la versión original del código de una función, causen que la función creer o se aproveche de una condición insegura. En muchos casos, tales ventajas pueden ser prevenidas usando tales parches para prevenir que la función se ejecute con tales valores de parámetro. En algunas personalizaciones, los parches especifican la comprobación de valores de otra manera que los parámetros de la función, tales valores leídos de un archivo o introducidos por el usuario.

[0023] En algunas personalizaciones, un agente remendando automatizado recibe cada parche automáticamente, lo valida, y lo almacena en una tabla de parches para la posible aplicación. En algunas personalizaciones, cada parche se aplica a cualquier instancia del módulo ejecutable a ser remendado que ya está cargado en el sistema de la computadora designada cuando el parche es recibido. Este acercamiento es referido aquí dentro como el "parche caliente – (Hot Fix) ", y permite a los parches ponerse eficazmente inmediatamente al recibirse, y no exige que la facilidad determine donde el módulo ejecutable a ser remendado se guarde en el disco. En algunas personalizaciones, cada parche recibido se aplica a la imagen del disco del módulo ejecutable a ser remendado, para que, cuando la imagen del disco sea cargada un tiempos futuros, la imagen del disco cargada incluya el parche. Este acercamiento referido aquí dentro como el "parche frío – Cold Fix", y permite a los parches ser persistente a través de sesiones múltiples. En algunas personalizaciones, la facilidad realiza el remendando caliente y frío. En algunas personalizaciones, cada parche se

aplica al módulo ejecutable a ser remendado cada vez que el módulo ejecutable a ser remendado es cargado por el cargador del sistema operativo. Este acercamiento es referido aquí dentro como "carga a tiempo del parche – load-time patching." En algunas personalizaciones, cada parche se aplica al módulo ejecutable a ser remendado cada vez que la función a ser remendada es invocada. Este acercamiento es referido aquí dentro como "llamada de interceptación de parche – call-interception patching." Ambas (1), carga a tiempo de parche y llamada a tiempo de parche no exige a la facilidad poder determinar donde el módulo ejecutable a ser remendado se guarda en el disco, (2) la facilidad de reversibilidad lista de un parche particular, y (3) no requiere modificación de la imagen del disco del módulo ejecutable.

[0024] En algunas personalizaciones, la facilidad permite a un usuario o a administrador configurar el funcionamiento de parches que han sido aplicados. Como ejemplos, tal configuración puede incluir, para un parche aplicado en particular: si la prueba especificada por el parche es realizada cuando la ejecución alcanza el punto especificado para el parche; si el resultado de la prueba del manejo especificado por el parche es realizado o ignorado; y/o si el rendimiento de la prueba y/o su resultado es autenticado, desplegado en un mensaje de advertencia, etc. En estas personalizaciones la facilidad permite probar los parches en los sistemas de computadora de producción habilitando y deshabilitando el manejo del resultado autenticado inicialmente. En estas personalizaciones, la facilidad más allá permite la operación del parche a ser autenticado en "un modo verboso" después de habilitar el manejo de su resultado para ayudar con identificar casos en los cuales el parche está creando un problema, como un problema de compatibilidad de aplicación u otro problema IT. Estas personalizaciones también permiten desactivar un parche rápidamente después de ser aplicado si se descubre que el parche está creando problemas. Algunas personalizaciones de la facilidad también permiten que el parche sea

13

desactivado rápidamente eliminando simplemente el parche del conjunto de parches recibido y guardado en el sistema de computadora designado.

[0025] En algunas personalizaciones, la facilidad usa un "manejo de datos – data-driven" acercamiento de parcheo, en el cual los parches no contienen ningún código, sino datos, como un texto pequeño humano-legible o documento de XML que especifican un punto al cual realizar una prueba, la identidad de la prueba a realizar, y cómo actuar en respuesta a uno o más resultados de prueba diferentes. En tales personalizaciones, un agente de parcheo recibe el manejo de datos (data-driven), y agrega las pruebas y el manejo de la prueba especificados por los parches. En algunas personalizaciones, la facilidad utiliza un acercamiento de parcheo de "manejo de código (code-driven)", y en el cual cada parche contiene un programa corto a ser agregado al módulo ejecutable a ser remendado que realiza la prueba llamando el parámetro normal de la facilidad que prueba las funciones, y que realiza el manejo de la prueba. Usando el parcheo de manejo de datos (data-driven) o el de manejo de código (code-driven), es posible algunas veces direccional todos los sabores de un módulo ejecutable a ser parchado con un solo parche.

[0026] En algunas personalizaciones, la facilidad autentica cada parche para demostrar ambos que (1) el parche es de una fuente aceptado, y (2) no se han modificado los volúmenes del parche desde el tiempo en que el parche se creó por la fuente aceptada.

[0027] En algunas personalizaciones, la facilidad distribuye cada parche para cada sistema de la computadora designada, y un agente de parches en el sistema de la computadora designada determina automáticamente cuales parches para ser aplicados en el sistema de la computadora designada, y cómo serán aplicados, basado en las características del sistema de la computadora designada. Esto releva a los usuarios y administradores de muchas de las cargas convencionalmente asociadas con seleccionar y aplicar los parches, y la carga de mantener una base de datos del inventario exacto y actual. Por ejemplo, estas características

pueden incluir qué versión del módulo ejecutable a ser remendado se instala en el sistema de la computadora designada. En estas personalizaciones, la facilidad puede superar el tipo de problemas causado por los parches calientes, distribuyendo parches que especifican el manejo diferente para los parches caliente y no calientes (hot-fixed y un-hot-fixed) de los diferentes sabores del módulo ejecutable en particular, obviando típicamente alguna necesidad de cualquier sacrificio de un parche caliente para un módulo ejecutable particular o hacer que el parche caliente sea ubicuo al remendar ese módulo ejecutable.

[0028] En algunas personalizaciones, el agente de parches almacena cada parche recibido en el sistema designado, independiente de si el módulo ejecutable a ser remendado por un parche particular se instala en el sistema designado cuando ese parche se recibe. Porque la facilidad en muchos casos aplica los parches en respuesta a la carga de un módulo ejecutable a ser remendado o la invocación de una función a ser remendada, la facilidad puede aplicar un parche a un módulo ejecutable que se instaló en el sistema designado después de que ese parche se recibió en el sistema designado. También, los parches pueden sobrevivir la desinstalación y la reinstalación subsiguiente del módulo ejecutable a ser remendado.

[0029] En algunas personalizaciones, el agente de parches es implementado en un servicio del sistema operativo. En estas personalizaciones, la facilidad otorga al agente de parches cualquier permiso exigido para aplicar un parche. Estas personalizaciones reducen el riesgo de seguridad impuesto típicamente cuando los parches convencionales son aplicados, cuando ellos obvian alguna necesidad para un usuario de autenticarse en el sistema de la computadora designada que usa una cuenta administrativa que tiene los permisos de modificación habilitados, y por eso dando a cualquier virus presente en el sistema de la computadora designada una oportunidad mayor de modificar aspectos sensibles del sistema de la computadora designada.

[0030] Los parches usados por la facilidad son típicamente relativamente pequeños, y por consiguiente imponen requisitos de recursos modestos para la transmisión y almacenamiento. También, porque los parches usados por la facilidad tienden a modificar el comportamiento del software remendado en un número pequeño de maneras bien-definidas, la facilidad ayuda a reducir el problema de código basura.

[0031] La figura 1 ilustra un ejemplo de un sistema de cómputo conveniente ambiente 100 en el cual la facilidad puede llevarse a cabo. El sistema de cómputo ambiente 100 es sólo un ejemplo de un ambiente de cómputo conveniente y no se piensa en cualquier limitación acerca del alcance de uso o funcionalidad de la facilidad. Tampoco debería ser interpretado el ambiente de cómputo 100 como si tuviera cualquier dependencia o requisito relacionado a cualquier combinación de componentes ilustrados en el ambiente operativo 100 ejemplar.

[0032] La facilidad es operacional con numerosos otros propósitos de propósito general o especial de los ambientes de cómputo del sistema o configuraciones. Los ejemplos de sistemas de cómputo bien conocidos, ambientes, y/o configuraciones que pueden ser convenientes para el uso con la facilidad incluyen, pero no se limita a: las computadoras personales, los servidores, los dispositivos portátiles o portables, los dispositivos de tabla (tablet devices), los sistemas multiprocesador, los sistemas basados en microprocesador, las cajas de juego, electrónicosa de consumo programables, red PCs, los miniordenadores, los sistemas informáticos grandes, ambientes de cómputo distribuidos que incluyen cualquiera de los sistemas anteriores o dispositivos, y similares.

[0033] La facilidad puede describirse en el contexto general de instrucciones de computadora-ejecutables, como los módulos de programa, ejecutándose por una computadora. Generalmente, los módulos de programa incluyen las rutinas, programas, objetos, componentes, estructuras de datos, y similares, qué realizan tareas particulares o implementan tipos de datos

abstractos particulares. La facilidad también puede practicarse en ambientes de la informática distribuídos dónde las tareas son realizadas por dispositivos del proceso remotos que se unen a través de una red de comunicaciones. En un ambiente de la informática distribuído, pueden localizarse los módulos del programa en los medios de comunicación de almacenamiento de computadora locales y/o remotos incluso los dispositivos de almacenamiento de memoria.

[0034] Con referencia a la Figurar 1, un sistema ejemplar para implementar la facilidad incluye un dispositivo de cómputo de propósito general en la forma de una computadora 110. Los componentes de la computadora 110 pueden incluir, pero no se limita a, una unidad de proceso 120, un sistema de memoria 130, y un sistema de bus 121 que se acopla a varios componentes del sistema incluso la memoria del sistema a la unidad de proceso 120. El sistema de bujs 121 puede ser cualquiera de varios tipos de estructuras de bus incluso un bus de memoria o controlador de memoria, un bus periférico, y un bus local que usan cualquiera de una variedad de arquitecturas de bus. Por vía del ejemplo, y sin limitación, tales arquitecturas incluyen el bus de la Industria de Arquitectura Estandar (ISA), el bus de Arquitectura de Micro Canal (MCA), el bus ISA extendido (EISA), el bus local de la Asociación de Estándares de Video Electrónico (VESA) el bus local, y el Componente de Interconección Periférico (PCI) también conocido como el bus del entresuelo.

[0035] La computadora 110 incluye una variedad de medios de comunicación típicamente legibles por computadora. Los medios de comunicación legibles por computadora pueden ser cualquier medios de comunicación disponible que puede accederse por la computadora 110 y puede ser incluidos ambos medios de comunicación volátil y no volátil, y los medios de comunicación removibles y no removibles. Por vía del ejemplo, y sin limitación, los medios de comunicación legibles por computadora pueden comprender medios de comunicación de almacenamiento de computadora y medios de comunicación. Los medios de comunicación de

almacenamiento de computadora incluyen volátil y no volátil, los medios de comunicación removibles y no removibles implementados en cualquier método o tecnología para el almacenamiento de información como las instrucciones legibles por computadora, las estructuras de datos, los módulos del programa u otros datos. Los medios de comunicación de almacenamiento de computadora incluyen, pero no se limita a, RAM, ROM, EEPROM, memoria de destello (flash memory) u otra tecnología de memoria, CD-ROM, los discos versátiles digitales (DVD) u otro almacenamiento de disco óptico, casetes magnéticos, cinta magnetofónica, almacenamiento de disco magnético u otros dispositivos de almacenamiento magnéticos, o cualquier otro medio que pueda usarse para guardar la información deseada y qué pueda ser accedida por la computadora 110. Los medios de comunicación incluyen las instrucciones legibles típicamente por computadora, las estructuras de datos, módulos de programa u otros datos en señales de datos moduladas como una onda portadora u otro mecanismo de transporte e incluyen cualquier medio de comunicación de entrega de información. El término "señal de datos modulada" significa que tiene uno o más de su conjunto de características o cambios de tal manera que codifica la información de la señal. Por vía del ejemplo, y sin limitación, los medios de comunicación incluyen los medios de comunicación alambrados como una red alambrada o conexión alambrada directa, y los medios de comunicación inalámbricos como el acústico, RF, infrarrojo y otros medios de comunicación inalámbricos. Las combinaciones de cualquiera de los anteriores también debe ser incluido dentro del alcance de medios de comunicación legibles por computadora.

[0036] El sistema de memoria 130 incluye los medios de comunicación de almacenamiento de computadora en la forma de memoria volátil y no volátil y/o tal como la memoria de solo lectura (ROM) 131 y la memoria de acceso aleatorio (RAM) 132. Un sistema básico de entrada/salida 133 (BIOS), conteniendo las rutinas básicas que ayudan a transferir la información entre los elementos dentro de la computadora 110, tal como durante el inicio, que

se guarda típicamente en la ROM 131. La RAM 132 típicamente contiene los datos y/o módulos del programa que son inmediatamente accesibles y/o presentemente siendo operados por la unidad de proceso 120. Por vía de ejemplo, y sin limitación, la figure 1 ilustra el sistema operativo 134, la aplicación del programa 135, otros módulos de programa 136 y programas de datos 137.

[0037] La computadora 110 también puede incluir otros medios de comunicación de computadora removibles y no removibles, volátiles y no volátiles. Por vía del ejemplo únicamente, la figure 1 ilustra un controlador de disco duro 141 que lee desde o escribe a un no removible, no volátil de los medios de comunicación magnéticos, una unidad de disco 151 magnética lee desde o escribe a un removible, no volátil disco 152 magnético, y una unidad de disco 155 óptica lee desde o escribe a un trasladable, no volátil disco 156 óptico como un CD ROM u otros medios de comunicación ópticos. Otro removible/no removible, volátil/no volátil medio de comunicación de almacenamiento de computadora que pueden usarse en el ambiente ejemplar que se opera incluyen, pero no se limita a, los casetes de cinta magnetofónica, tarjetas de memoria de destello, discos versátiles digitales, cinta de video digital, RAM de estado sólida, ROM de estado sólido, y similares. La unidad de disco duro 141 se conecta típicamente al sistema de bus 121 a través de una memoria non-removible como la interfaz 140, y la unidad de disco magnética 151 y la unidad de disco óptico 155 son típicamente conectados al sistema de bus 121 por una interfase de memoria removible 150.

[0038] Los controladores y sus medios de comunicación de almacenamiento de computadora asociados, discutidos anteriormente e ilustrados en la figura 1, proporcionan almacenamiento de instrucciones legibles por computadora, las estructuras de datos, los módulos de programa y otros datos para la computadora 110. En la figura 1, por ejemplo, el controlador del disco duro 141 es ilustrado como sistema operativo de almacenamiento 144, los programas de aplicación 145, otros módulos de programa 146 y

programas de datos 147. Note igual que estos componentes pueden ser diferentes o los mismos del sistema operativo 134, el programa de aplicación 135, otros módulos de programa 136, y datos de programa 137. El sistema operativo 144, el programa de aplicación 145, otros módulos de programa 146, y datos de programa 147 a los cuales son dados aquí dentro números diferentes para ilustrar que, en un mínimo, ellos son copias diferentes. Un usuario puede entrar las órdenes e información en la computadora 110 a través de los dispositivos de entrada como tabla, o digitalizador electrónico, 164, un micrófono 163, un teclado 162 y un dispositivo de apuntamiento 161, normalmente llamado ratón, trackball o almohadilla de tacto. Otros dispositivos de entrada no mostrados en la figura 1 pueden incluir una palanca de mando, la almohadilla de juego, plato satélite, escáner, o similares. Se conectan a menudo éstos y otros dispositivos de entrada a la unidad de proceso 120 a través de una interfaz de entrada de usuario 160 que se acopla al bus del sistema, pero puede conectarse por otra interfaz y estructuras de bus, como un puerto paralelo, un puerto de juego o un bus de serie universal (USB). Un monitor 191 u otro tipo de dispositivo de despliegue también se conecta al sistema de bus 121 vía una interfaz, como una interfaz de video 190. El monitor 191 también puede integrarse con una pantalla de tacto o similares. Note que el monitor y/o el tablero de pantalla de tacto puede acoplarse físicamente a un alojamiento en el cual el dispositivo de cómputo 110 está incorporado, como en un computador personal tipo tabla. Además, las computadoras como el dispositivo de cómputo 110 también puede incluir otros dispositivos de rendimiento periféricos como parlantes 195 e impresora 196 que pueden conectarse a través de una interfaz periférica de salida 194 o similares.

[0039] La computadora 110 puede operar en un ambiente conectado a una red de computadoras que usa conexiones lógicas o a las computadoras más remotas, como una computadora remota 180. La computadora 180 remota puede ser una computadora personal, un servidor, un enrutador, una red de PCs, un dispositivo par u otro nodo de la red común, y

típicamente incluye muchos o todos los elementos descritos arriba relacionados a la computadora 110, aunque sólo un dispositivo de almacenamiento de memoria 181 se ha ilustrado en la figura 1. Las conexiones lógicas pintadas en la figura 1 incluyen una red de área local (LAN) 171 y una red de área amplia (WAN) 173, pero también puede incluir otras redes. Tales ambientes de gestión de redes son comúnes en las oficinas, en las redes de computadores de empresas grandes, intranets e Internet. Por ejemplo, en la facilidad presente, el sistema de cómputo 110 puede comprender la fuente de máquina de la cual los datos están siendo migrados, y la computadora remota 180 puede comprender la máquina del destino. Note sin embargo que esas máquinas fuente y destino no necesita ser conectados por una red o cualquier otro medio, pero en cambio, pueden emigrarse los datos por cualquier medio de comunicación capaz de ser escrito por la plataforma de fuente y pueden leerse por la plataforma o plataformas destino.

[0040] Cuando es usado en un ambiente de red LAN que conecta una red de computadoras, la computadora 110 se conecta a la LAN 171 a través de una interfaz de red o adaptador 170. Cuando es usado en un ambiente de redes WAN, la computadora 110 incluye un módem 172 u otro medio típicamente para establecer las comunicaciones sobre la WAN 173, como Internet. El módem 172 puede ser interno o externo y puede conectarse al sistema de bus 121 vía la interfaz de usuario 160 u otro mecanismo apropiado. En un ambiente de red conectado, los módulos de programa graficados relacionados a la computadora 110, o parte de ello, puede guardarse en el dispositivo de almacenamiento de memoria remoto. Por vía de ejemplo, y sin limitación, la figure 1 ilustra el programa de aplicación remota 185 como residente en el dispositivo de memoria 181. Se apreciará que las conexiones de red mostradas son medios ejemplares y otros medios para establecer un eslabón de comunicaciones entre las computadoras pueden usarse.

[0041] Mientras varias funcionalidades y datos son mostradas en la figura 1 como residentes en sistemas de cómputo particulares que son colocados de una manera particular, estos experimentados en el arte apreciarán que tales funcionalidades y datos pueden distribuirse de varias otras maneras por los sistemas de cómputo en los diferentes arreglos. Mientras los sistemas de cómputo configurados como los descritos anteriormente son típicamente usados para apoyar el funcionamiento de la facilidad, una habilidad ordinaria en el arte apreciará que la facilidad puede ser implementada usando dispositivos de varios tipos y configuraciones, y teniendo varios componentes.

[0042] La figura 2 es un diagrama de flujo de datos que muestra un intercambio típico de datos entre los sistemas de cómputo de acuerdo con la facilidad. Los sistemas de cómputo mostrados en la figura 2 (sistemas de cómputo 210, 220, 221, 222, 230, 231, y 232) típicamente tienen algunos o todos los componentes mostrados y discutidos junto con la figura 1. En el servidor de distribución de parches, la facilidad genera uno o más parches. Estos parches 201 se envían del servidor de distribución de parche a uno o más servidores de administración, como los servidores de administración 220 y 230. Cada servidor de administración, a su vez, adelanta los parches a uno o más sistemas de cómputo designados, como la computadora designada sistemas 221, 222, 231, y 232. En algunas personalizaciones (no mostradas), el servidor de distribución de parches envía los parches directamente a uno o más sistemas de cómputo designados, vía una ruta más indirecta que a través de un solo servidor de administración. Se procesan los parches recibidos en un sistema de cómputo designado como descrito en mayor detalle debajo. Un servidor de administración también puede enviar comandos de la configuración del parche 202 a uno o más sistemas . de cómputo designados que aplican los comandos de configuración del parche que reconfiguran la operación de parches particulares. Como se discute en mayor detalle debajo, un parche puede desactivarse completamente; si un parche no es inválido, la notificación de

su funcionamiento y su manejo de resultado de prueba puede habilitarse o deshabilitarse a cada uno independientemente. Cuando la notificación de su funcionamiento se habilita, pueden desplegarse las notificaciones o pueden guardarse localmente en el sistema de la computadora designada, o pueden enviarse como notificaciones 203 a un servidor de administración apropiado.

[0043] La figura 3 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para recibir y procesar un nuevo parche. En el paso 301, la facilidad recibe un parche. El parche recibido en el paso 301 puede ser un parche de manejo de datos o un parche de manejo de código. Una muestra de parche de código de datos se muestra en la tabla 1 debajo.

```
1    <Softpatch Patch="Q382429">
2     <AffectedApplication AffectedExe="sqlservr.exe">
3      <AffectedVersion Version="9.*">
4       <AffectedModules Name="SQLSORT.DLL">
5        <Version "8.0.*, 9.*">
6              <Function Name="SsrpEnumCore" Address="0x0802E76B"
7                  Param="2" Paramtype="LPSTR">
8               <Filter MaxByteLength="60" />
9               <Resolution ActionType="BOOL" Action="FALSE" />
10             </Function>
11       </Version>
12       <Version "10.*, 11.*">
13             <Function Name="SsrpEnumCore" Address="0x0802D283"
14                 Param="2" Paramtype="LPSTR">
15              <Filter MaxByteLength="128" />
16              <Resolution ActionType="BOOL" Action="FALSE" />
17             </Function>
18       </Version>
19      </AffectedModules>
20     </AffectedVersion>
21    </AffectedApplication>
22
23    <Signature Hash="MD5" Signature="C509-04AA-9161-8C52-
24     9F6D-BF5A-AEF2-ECE1-0038-34D1"/>
25   </Softpatch>
```

TABLA 1

[0044] La línea 1 contiene un único identificador para el parche. La línea 2 identifica la aplicación afectada por el parche. La línea 3 identifica la versión de la aplicación afectada por el parche. La línea 4 identifica el módulo ejecutable afectado por el parche. La línea 5 identifica dos versiones del módulo ejecutable afectado–versiones 8.0. * y 9. *– para los cuales las direcciones del parche son proporcionados. Las líneas 6-10 contienen las direcciones de parche para estas dos versiones del módulo ejecutable. Las líneas 6-7 identifican la función a ser remendada, su dirección en el módulo ejecutable, su parámetro a ser probado por el parche, y el tipo del parámetro a ser probado. La línea 8 indica que el parámetro identificado en las líneas 6-7 debe probarse para determinar si su longitud excede 60 bytes. La línea 9 indica que la invocación de la función debe fallar si la prueba tiene éxito. La línea 12 identifica dos ó más versiones del módulo ejecutable afectado – las versiones 10. * y 11. *– para las cuales las direcciones del parche se proporcionan. Las líneas 13-17 contienen las direcciones del parche para estas dos versiones del módulo ejecutable. Las líneas 13-14 identifican la función a ser remendada, su dirección en el módulo ejecutable, su parámetro a ser probado por el parche, y el tipo del parámetro a ser probado. Puede verse que la dirección de la función a ser remendada en el módulo ejecutable identificado en las líneas 13-14 para versiones 10. * y 11. * es diferente de la dirección de la función a ser remendada identificada en las líneas 6-7 para las versiones 8.0. * y 9. *. La línea 15 indica que el parámetro identificado en las líneas 13-14 que debe probarse para determinar si su longitud excede 128 bytes. La línea 16 indica que la invocación de la función debe fallar si la prueba tiene éxito. El parche puede especificar una variedad de tipos de acción de manejo de resultado, incluso el fracaso de la invocación de la función remendada, levantando una excepción, terminando el proceso en que el módulo ejecutable remendado está ejecutándose, o corrigiendo el valor

erróneo (tal como truncando una cadena demasiado larga). Las líneas 23-25 contienen una firma para el parche que identifica la fuente del parche y verifica que el parche no ha sido alterado desde de su fuente.

[0045] La Tabla 2 debajo contiene una versión de manejo de código (code-driven) del parche mostrado anteriormente en la Tabla 1.

```
1     00411A7E  push     3Ch
2     00411A80  mov      eax,dword ptr [str]
3     00411A83  push     eax
4     00411A84  call     ValidateStringLength (411082h)
5     00411A89  add      esp,8
6     00411A8C  movzx    ecx,al
7     00411A8F  test     ecx,ecx
8     00411A91  je       411A9Ah
9     00411A93  jmp      foo+2 (411AD2h)
10    00411A9A  xor      eax,eax
11    00411A9C  ret
```

TABLA 2

[0046] Las líneas 1-3 empujan los parámetros para la función de prueba hacia la pila. La línea 4 llama la función de prueba. Las líneas 5-8 dependen del código de retorno para la función de comprobación. Si la función de prueba tuviera éxito, la línea 9 retorna para empezar ejecutando el cuerpo de la función remendada. Si la función de prueba fallara, las líneas 10-11 empujan un código de resultado de fracaso hacia la pila y retorna desde la función de reparcheo al color de la función remendada. Para la legibilidad, la Tabla 2 omite ciertos detalles presentes en algunos parches de manejo de código (code-driven), incluso una firma comprobable, las instrucciones que prueban para los valores actuales de las banderas de configuración del parche, y las instrucciones relocalizadas desde el principio del código para la función de reparcheo.

[0047] En algunas personalizaciones, los parches de ambos tipos pueden contener la información adicional, incluyendo, para cada uno de una o más versiones del módulo ejecutable para ser parchado, una firma de archivo que puede usarse para validar que un caso particular del módulo ejecutable es una copia apropiada de esa versión. Tal archivo de firma puede ser, por

ejemplo, un tamaño o una verificación para la versión del módulo ejecutable completo, o el código esperado a ocurrir en un punto particular en el módulo ejecutable, como el desplazamiento al que el módulo ejecutable será remendado.

[0048] En el paso 302, si el parche se firma con una firma válida, entonces la facilidad continúa al paso 303, el resto de la facilidad continúa al paso 301 para recibir el próximo parche. En el paso 303, la facilidad agrega el parche a una tabla del parche local. En el paso 304, la facilidad inicializa la configuración inicial del parche, como enviándolo a una configuración predefinida.

[0049] La figura 4 es un diagrama de estructura de datos que muestra una tabla de parche de muestra típico de aquéllos usados por la facilidad. La tabla del parche 400 contiene las filas, como las filas 401 y 402, cada una dividida en las columnas siguientes: una columna identificadora del parche 411, conteniendo el identificador del parche extraído del parche,; una columna del módulo ejecutable 412, conteniendo la información que identifica el módulo ejecutable a ser reparchado, tal como su nombre,; una columna de versiones de módulo ejecutables 413 que identifica todas las versiones del módulo ejecutable identificado en la columna 412 al que el parche aplica; una columna habilitada de rendimiento de prueba 414, conteniendo el valor de la configuración actual cual por si la prueba especificada por el parche debería realizarse cada vez que la función de reparchado es llamada; una columna habilitada de notificación de rendimiento de prueba 415, conteniendo el valor de la configuración actual por si una notificación debe generarse cada vez que la prueba del parche se ha realizado; una columna habilitada de notificación de resultado de prueba 416, conteniendo el valor de la configuración actual por si una notificación debería generarse cada vez que la prueba del parche tiene éxito; una columna habilitada de manejo de resultado de la prueba 417, conteniendo el valor de la configuración actual por si el manejo de resultado del parche debe llevarse a cabo cuando fallan las pruebas del parche; y una

columna del parche 418 que contiene un apuntador al propio parche, especificando una prueba y un manejo de resultado de la prueba para ser realizado cada vez que falla la prueba. En algunas personalizaciones, en lugar de contener un indicador como el parche mostrado, la columna del parche 418 contiene cada parche directamente. Una tabla de parche particular puede contener o puede apuntar a parches de varios tipos, como todos los parches de manejo de código, todos los parches de manejo de datos, o una combinación de parches de manejo de código y de manejo de datos.

[0050] En el paso 305, una vez la facilidad ha agregado el parche recibido a la tabla de parches y ha inicializado sus configuraciones, el parche está disponible para ser aplicado automáticamente por la facilidad en el módulo ejecutable. En el paso 305, la facilidad puede utilizar una variedad de acercamientos para aplicar el parche, incluyendo aquéllos descritos en la aplicación incorporada por la referencia, así como la función de tiempo real llamada interceptación, y/o la reescritura de código de (1) módulos ejecutables ya listos cargados, (2) uno o más imágenes del módulo ejecutable del disco, o (3) instancias de módulos ejecutables cargados por el cargador del sistema operativo. Después del paso 305, la facilidad continúa en el paso 301 para recibir el próximo parche.

[0051] La figura 5 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para poner al día las instrucciones de la configuración para un parche en particular. En el paso 501, la facilidad recibe las instrucciones de la configuración para un parche en particular, tal como de un administrador. En algunas personalizaciones, tales instrucciones de configuración pueden ser generadas por administradores que usan las políticas de grupo. En el paso 502, la facilidad actualiza que el parche es la configuración de la Tabla de parches de acuerdo con las instrucciones recibidas. Después del paso 502, la facilidad continúa en el paso 501 para recibir las próximas instrucciones de configuración.

[0052] La figura 6 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para realizar la aprobación del parámetro especificado por un parche. En el paso 601, una función de parcheo es llamada. En el paso 602, si la prueba es habilitada para el parche que afecta la función llamada, entonces la facilidad continúa en el paso 603, sino, la facilidad continúa en el paso 601 para procesar la próxima llamada a una función de parcheo. En el paso 603, si la notificación de rendimiento de prueba se habilita para el parche, entonces la facilidad continúa en el paso 604, sino la facilidad continúa en el paso 605. En el paso 604, la facilidad genera una notificación de que la prueba fue realizada. En los pasos 604, 608, y 610 pueden involucrar el despliegue o almacenamiento de una indicación en el sistema de la computadora designada de que la prueba fue satisfecha, y/o transmitiendo tal indicación a un sistema de cómputo remoto para el despliegue o registro allí.

[0053] En el paso 605, la facilidad realiza la prueba de aprobación especificada por el parche. En algunas personalizaciones, el paso 605 involucra la llamada de un grupo de rutinas normales utilizada spor la facilidad para probar. En el paso 606, si la prueba realizada en el paso 605 fue satisfecha, entonces la facilidad continúa en el paso 601, sino la facilidad continúa en el paso 607. En el paso 607, si la notificación de resultado de prueba es habilitado para el parche, entonces la facilidad continúa en el paso 608, sino la facilidad continúa en el paso 609. En el paso 608, la facilidad genera una notificación de que la prueba no fue satisfecha. En el paso 609, si el manejo del resultado de la prueba se habilita para el parche, entonces la facilidad continúa en el paso 610, sino, la facilidad continúa en paso 601. En paso 610, la facilidad realiza el prueba resultado manejando especificó por el parche. Después del paso 610, la facilidad continúa en el paso 601.

[0054] Será apreciado por estos experimentado en el arte que la facilidad arriba descrita puede adaptarse correctamente o puede extenderse de varias maneras. Por ejemplo, la facilidad puede usarse para aplicar una

variedad de tipos diferentes de parches en una variedad de maneras a una variedad de situaciones en los módulos ejecutables de una variedad de tipos para una variedad de propósitos. También, mientras son descritos los parches aquí dentro de cómo contener pruebas de aprobación de valor que indican un problema cuando ellos fallan, la facilidad también puede ser implementada usando la pruebas de invalidez de valor que indican un problema que puede llevarse a cabo cuando ellos tienen éxito. En algunas personalizaciones, cada prueba se acompaña por una indicación de si el éxito o el fracaso indica un problema. Mientras la descripción anterior hace referencia a las personalizaciones preferidas, el alcance de la invención se define solamente por las demandas que siguen y los elementos citados aquí dentro.

LAS DEMANDAS

Nosotros Demandamos:

[c1] 1. Un método en un sistema de cómputo para aumentar el software, comprendiendo:

Recibir en un sistema de cómputo designado una especificación de aumento que especifica (a) una función a ser aumentada, la función especificada estando entre el software ejecutado en el sistema de cómputo, (b) un parámetro de la función a ser aprobado, (c) una prueba para aplicar al parámetro especificado, y (d) una modificación para realizar al comportamiento de función si la prueba especificada no es satisfecha por el parámetro especificado; y

Cuando la función especificada es invocada en el sistema de la computadora designada, si la prueba especificada no es satisfecha por el parámetro especificado, realizar la modificación específica a la función especificada.

[c2] 2. El método de la demanda 1 en donde la modificación especificada por la especificación de aumento está previniendo la ejecución de la función especificada.

[c3] 3. El método de la demanda 1 de la modificación especificada por la especificación de aumento está ejecutando la función especificada después de la alteración del parámetro especificado.

[c4] 4. El método de la demanda 1, en donde una pluralidad de especificaciones de aumento se recibe en el sistema de la computadora designada.

[c5] 5. El método de la demanda 4, más allá comprende el manteniendo de todas las especificaciones de aumento recibidas en una estructura de datos.

[c6] 6. El método de la demanda 1 en qué ninguna intervención de usuario subsiguiente se requiere para recibir la especificación de aumento para realizar la modificación especificada a la conducta de la función especificada.

[c7] 7. El método de la demanda 1 en donde la especificación de aumento especifica una prueba para aplicar al parámetro especificado identificando un parámetro que prueba la función para invocar de quien el código no es incluido en la especificación de aumento.

[c8] 8. El método de la demanda 1 en donde la especificación de aumento especifica una prueba a ser aplicada al parámetro especificado identificando un parámetro que prueba la función para invocar que fue instalado antes de la especificación de aumento recibida.

[c9] 9. El método de la demanda 1 en donde la especificación de aumento contiene el código.

[c10] 10. El método de la demanda 9 en donde el código contenido por la especificación de aumento incluye código que invoca una función de prueba de parámetro cuyo código no es incluido en la especificación de aumento.

[c11] 11. El método de la demanda 1 en donde la especificación de aumento contiene texto que identifica una función de prueba de parámetro cuyo código no está incluido en la especificación de aumento.

[c12] 12. El método de la demanda 11 en donde una función de prueba de parámetro identificada por el texto contenido por la especificación de aumento está contenida por la especificación de aumento.

[c13] 13. El método de la demanda 11 en donde una función de prueba de parámetro identificada por el texto no contenido por la especificación de aumento no está contenida por la especificación de aumento.

[c14] 14. El método de la demanda 1 más allá comprende proporcionar una interfaz para configurar un estado para controlar la operación de la especificación de aumento,
y en donde la modificación especificada para el comportamiento de la función especificada es realizada solo cuando el estado es un estado habilitado.

[c15] 15. El método de la demanda 14 en donde la interfaz proporcionada permite que la operación de la especificación de aumento sea configurada localmente.

[c16] 16. El método de la demanda 14 en donde en donde la interfaz proporcionada permite la operación de la especificación de aumento sea configurada remotamente.

[c17] 17. El método de la demanda 14 en donde donde la interfaz proporcionada permite la operación de la especificación de aumento sea configurada de acuerdo con una política.

[c18] 18. El método de la demanda 14 en donde la prueba especificada es realizada antes de que cualquier código sustantivo de la función especificada se ejecute.

[c19] 19. El método de la demanda 1 más allá comprende proporcionar una interfaz para la configurando de advertencias asociadas con la especificación de aumento, y donde, si la interfaz proporcionada es usada para habilitar las advertencias, generando una advertencia cada vez que la prueba especificada por la especificación de aumento falla.

[20] 20. El método de la demanda 1, más allá comprende proporcionar una interfaz para configurar las notificaciones asociadas con la especificación de aumento, y en donde, si la interfaz proporcionada se usa para habilitar las notificaciones, generando una notificación cada vez que la prueba especificada por la especificación de aumento es realizada.

[c21] 21. El método de la demanda 20, más allá comprende consolidar las notificaciones generadas en una sola notificación consolidada.

[c22] 22. El método de la demanda 1 en donde la especificación de aumento recibida es firmada,
el método más allá comprende verificar la firma de la especificación de aumento,

y en donde la modificación especificada para el comportamiento de la función especificada sólo es realizado si la verificación de la firma de la especificación de aumento tuvo éxito.

[c23] 23. El método de la demanda 22 en donde la modificación especificada para el comportamiento de la función especificada es realizado solo si el autenticador de la firma de la especificación de aumento coincide con un autenticador del software que contiene la función especificada.

[c24] 24. El método de la demanda 1, más allá comprende realizar la prueba especificada en respuesta a descubrir que la función especificada ha sido invocada, sin alterar el código de la función especificada.

[c25] 25. El método de la demanda 1, más allá comprende alterar el código de la función especificada para realizar la prueba especificada cuando la función especificada es invocada.

[c26] 26. El método de la demanda 25 en donde el código de la función especificada es alterada insertando en el código de la función especificada un salto a una rutina separada que realiza la prueba especificada.

[c27] 27. El método de la demanda 1 en donde la función especificada es proporcionada en un módulo ejecutable distinguido que es posible cargar usando a un cargador, el método más allá comprende registrar con el cargador para tener una rutina que realiza el llamado de la prueba especificada antes de que la función especificada sea ejecutada en respuesta a cada invocación de la función especificada.

[c28] 28. Un sistema de cómputo que habilita el aumento de software presente en el sistema de cómputo, comprendiendo:

un dispositivo de almacenamiento que contiene el software;

un receptor del parche que recibe en el sistema de cómputo un parche especificando (a) un punto al cual el software será aumentado, (b) un valor asociado con la función a ser probada en el punto especificado, (c) una prueba para aplicar al valor especificado, y (d) una modificación para realizar al comportamiento del software si la prueba especificada no es satisfecha por el valor especificado; y

un agente parches que inyecta el código en el software al punto especificado tal que, cuando la ejecución del software alcanza el punto especificado en el sistema de la computadora designada, si la prueba especificada no es satisfecha por el valor especificado, la modificación especificada al comportamiento del software es realizada.

[c29] 29. Un medio legible por computadora cuyos contenidos causan a un sistema de cómputo designado para realizar un método para adicionar la aprobación de valor del software disponible en el sistema de cómputo designado, comprendiendo:

recibir en el sistema de cómputo una especificación de aumento que especifica (a) una función para la cual la aprobación de valor será agregada, la función especificada que está entre el software disponible y el sistema de cómputo, (b) los datos a ser probados que existen en el sistema de cómputo designado durante la ejecución de la función, (c) una prueba para aplicar a los datos especificados, y (d) una modificación para realizar al comportamiento de la función si la prueba especificada no es satisfecha por los datos especificados; y

cuando la función especificada se invoca en el sistema de la computadora designada, si la prueba especificada no es satisfecha por los datos especificados, realizando la modificación especificada al comportamiento de la función especificada.

[c30] 30. El medio legible por la computadora de demanda 22 en donde la prueba especificada es una prueba que siempre es satisfecha, independientemente del valor de los datos especificados.

[c31] 31. Una señal de datos generada llevando un parche de la estructura de datos, comprendiendo:

información que identifica una función a la cual la aprobación de valor será agregada;

información que identifica datos a ser probados que existen durante la ejecución de la función;

información que identifica una prueba para aplicar a los datos especificados; y para

información que identifica una modificación para realizar al comportamiento de la función si la prueba especificada no es satisfecha por los datos especificados,

Así, si la señal de datos es recibida en un sistema de la computadora designada en el cual la función es ejecutada, los contenidos de la estructura de datos puede ser usada para realizar la modificación especificada al comportamiento de la función especificada si la prueba especificada no es satisfecha por los datos especificados cuando la función especificada se invoca en el sistema de la computadora designada.

[c32] 32. La señal de datos generada de la demanda 31 en donde la señal de datos generada se transmite entre los sistemas de cómputo.

[c33] 33. La señal de datos generada de la demanda 31 en donde La señal de datos generada se transmite dentro de un sistema de cómputo.

[c34] 34. Un método para agregar la aprobación de valor al software disponible en el sistema de cómputo designado, comprendiendo:

recibir en el sistema de cómputo una especificación de aumento que especifica (a) software para el cual la aprobación de valor será agregada, (b) un punto en el software especificado en el cual la aprobación de valor será agregada, (c) los datos a ser probados que existen en el sistema de la computadora designada durante la ejecución del software especificado en

el punto especificado, (d) una prueba para aplicar a los datos especificados, y (e) una modificación para realizar al comportamiento del software si la prueba especificada no está satisfecha por los datos especificados;

permitir a un usuario que configure un modo de operación para la especificación de aumento recibida; y

cuando el software especificado se ejecute en el punto especificado en el sistema de la computadora designada, aplicando la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida.

[c35] 35. El método de la demanda 34 en donde el usuario configura un modo operacional activo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende, si la prueba especificada no está satisfecha por los datos especificados, realizando la modificación especificada al comportamiento del software especificado.

[c36] 36. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico activo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende, si la prueba especificada no está satisfecha por los datos especificados, ambos (1) realizar la modificación especificada al comportamiento del software especificado, y (2) generar una notificación de que la prueba especificada no es satisfecha por los datos especificados.

[c37] 37. El método de la demanda 34 en donde el usuario configura un modo operacional activo difuso para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

si la prueba especificada no es satisfecha por los datos especificados, realizando la modificación especificada al comportamiento del software especificado; y

generar una notificación de que la prueba especificada fue realizada.

[c38] 38. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico difuso activo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

si la prueba especificada no es satisfecha por los datos especificados, ambos (1) realizar la modificación especificada a la conducta del software especificado, y (2) generar una notificación de que la prueba especificada no es satisfecha por los datos especificados; y

generar una notificación de que la prueba especificada fue realizada.

[c39] 39. El método de la demanda 34 en donde el usuario configura un modo operacional inactivo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados.

[c40] 40. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico inactivo I para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados; y

si la prueba especificada no está satisfecha por los datos especificados, generando una notificación de que la prueba especificada no es satisfecha por los datos especificados.

[c41] 41. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico inactivo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados; y

generar una notificación de que la prueba especificada fue realizada.

[c42] 42. El método de la demanda 34 en donde el usuario configura modo operacional de diagnóstico inactivo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados;

generar una notificación de que la prueba especificada fue realizada; y si la prueba especificada no es satisfecha por los datos especificados, generando una notificación de que la prueba especificada no es satisfecha por los datos especificados.

ABSTRACTO

Una facilidad para el software de aumento en un sistema de cómputo designado es descrito. La facilidad recibe y la especificación de aumento en el sistema de la computadora designada. La especificación de aumentos especifica: (a) una función a ser aumentada, (b) un parámetro de la función a ser probado, (c) una prueba para aplicar al parámetro especificado, y (d) y la modificación para realizar al comportamiento de la función si la prueba especificada no es satisfecha por el parámetro especificado. Cuando la función especificada es invocada en el sistema de la computadora designada, si lo probado especificado no es satisfecho por el parámetro especificado, la facilidad realiza la modificación especificada al comportamiento de la función especificada.

SEGUNDA REIVINDICACIÓN

Reivindicamos:

[c43] 1. Un método de un sistema de computación que incluye:
recepción de un paquete de parches distintivos para modificar el comportamiento de un programa instalado en un sistema de computación;
la extracción automática de la información de la aplicación de dicho paquete de parches distintivos (1), que identifica una porción distintiva de un programa distintivo al cual se aplica el parche, y (2) información del comportamiento del parche que especifica la manera en la que se modifica el comportamiento de la porción del programa distintivo; y
la adición automática de una entrada distintiva a una tabla de parches, la cual incluye la información de la aplicación del parche de extracción y la información del comportamiento del parche.
[c44] 2. El método de la reivindicación 1, que además incluye la utilización de contenidos de la entrada distintiva para modificar el comportamiento de la porción distintiva del programa distintivo.
[c45] 3. El método de la reivindicación 2, por el que los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva del programa distintivo en un momento en que ningún usuario administrativo se encuentre utilizando el sistema de computación.
[c46] 4. El método de la reivindicación 2, por el que los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva del programa distintivo sin necesidad de autorización alguna de usuario.
[c47] 5. El método de la reivindicación 2, por el que los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva del programa distintivo sin determinar la ubicación donde dicho programa distintivo se guarda de manera persistente.
[c48] 6. El método de la reivindicación 2, por el que los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva del

programa distintivo como respuesta de la instalación de dicho programa distintivo.

[c49] 7. El método de la reivindicación 2, por el que los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva del programa como respuesta a la ejecución de la porción distintiva del programa distintivo.

[c50] 8. El método de la reivindicación 2, por el que se instalan dos instancias del programa distintivo, y los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva de las dos instancias del programa distintivo.

[c51] 9. El método de la reivindicación 1, que además incluye:

la determinación si el programa distintivo está instalado en el sistema de computación al momento en que se extrae la información de la aplicación del parche; y

si el programa distintivo está instalado en el sistema de computación, utilizando la información extraída de la aplicación del parche para modificar el comportamiento del programa distintivo instalado de acuerdo con la información del comportamiento del parche.

[c52] 10. El método de la reivindicación 1, que además incluye la eliminación de la entrada distintiva de la tabla del parche para prevenir que el comportamiento de la porción distintiva del programa distintivo se modifique de acuerdo con la información del comportamiento del parche contenida en la entrada distintiva.

[c53] 11. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de un valor.

[c54] 12. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de parámetros.

[c55] 13. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la

porción distintiva del programa distintivo para llevar a cabo la validación de un valor leído de un archivo.

[c56] 14. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de un valor ingresado por el usuario.

[c57] 15. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de un valor recibido de uno o más paquetes de la red.

[c58] 16. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de parámetros por medio de la utilización de una función de validación distintiva.

[c59] 17. El método de la reivindicación 16, por el que una entrada en la tabla de parches distinta de la entrada distintiva, también incluye la información del comportamiento del parche que especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de parámetros utilizando la función de validación distintiva.

[c60] 18. El método de la reivindicación 16, que además incluye el código de ejecución por el cual la función de validación distintiva modifica el comportamiento de la porción distintiva del programa distintivo, siendo que este código de ejecución no está incluido dentro del paquete de parches distintivos.

[c61] 19. Un medio reconocido por el computador, cuyo contenido lleve a que el sistema de computación realice un método que comprende:

la recepción de un paquete de parches distintivos que modifican el comportamiento de una entidad de programación en un sistema de computación;

la extracción automática de la información de la aplicación de dicho paquete de parches distintivos (1), que identifica una entidad de programación distintiva al cual se aplica el parche, y (2) información del comportamiento del parche que especifica la manera en la que se modifica el comportamiento de la porción de la

entidad de programación; y

la adición automática de una entrada distintiva a una tabla de parches, la cual incluye la información de la aplicación del parche de extracción y la información del comportamiento del parche.

[c62] 20. El medio reconocido por el computador de la reivindicación 19 por el cual la entidad de programación distintiva tiene varios comportamientos, de los cuales sólo un subconjunto adecuado está incluido en la información de comportamiento del parche que especifica la modificación, y por el cual el paquete de parches distintivos recibidos no contiene información sobre los comportamientos de la entidad de programación distintiva que la información del comportamiento del parche no especifica la modificación.

[c63] 21. Una o más memorias de computador que guardan de manera colectiva una estructura de de información de tabla de parche, que incluye varias entradas de parche, cada una de las cuales contiene:

información de la aplicación del parche, que identifica una porción distintiva de una entidad de programación distintiva al cual se debe aplicar el parche; e

información del comportamiento del parche, que especifica la manera en que se debe modificar el comportamiento de la porción distintiva de la entidad de programación distintiva, tal que, para una entrada de parche dada, los contenidos de dicha entrada pueden ser utilizadas para modificar el comportamiento de la porción distintiva de la entidad de programación distintiva en la manera especificada.

[c64] 22. Las memorias de computador de la reivindicación 21, por las cuales la información de la aplicación del parche identifica un módulo ejecutable asociado con la entidad de programación.

[c65] 23. Las memorias de computador de la reivindicación 22, por las cuales la información de la aplicación del parche identifica una posición en el módulo ejecutable identificado.

[c66] 24. Las memorias de computador de la reivindicación 23, por las cuales la información de la aplicación del parche identifica los contenidos esperados en la posición identificada en el módulo ejecutable identificado.

AN

[c67] 25. Las memorias de computador de la reivindicación 21, por las cuales una entrada de parche seleccionada contiene varias instancias de información de aplicación del parche, cada una de las cuales identifica una porción distintiva de una versión diferente de la entidad de programación distintiva.

[c68] 26. Las memorias de computador de la reivindicación 25, por las cuales una primera instancia de información de aplicación del parche identifica una versión de la entidad de programación distintiva al cual se aplicó una situación activa, y por el cual una segunda instancia de información de aplicación del parche identifica una versión de la entidad de programación distintiva al cual no se aplicó una situación activa.

[c69] 27. El método de la reivindicación 26, por el cual la entrada del parche seleccionado puede ser aplicado a (1), la versión de la entidad de programación distintiva al cual se aplica una situación activa, (2) la versión de la entidad e programación distintiva al cual no se aplica la situación activa seleccionada, o (3) ambos, sin reemplazar la versión de la entidad de programación distintiva al cual se aplica la situación activa seleccionada con la versión de la entidad de programación distintiva al cual no se aplica la situación activa, y sin reemplazar la versión de la entidad de programación distintiva al cual no se ha aplicado la situación activa con la versión de la entidad de programación distintiva al cual se aplicó la situación activa seleccionada; y

[c70] 28. El método de la reivindicación 24, por el cual la entrada del parche seleccionado puede ser desplegada a un sistema de computación objetivo por un administrador, sin importar qué versión de la entidad de programación distintiva se ejecute en el sistema de computación objetivo.

[c71] 29. El método de la reivindicación 21, por el cual la información del comportamiento del parche de una entrada de parche seleccionada contiene un código utilizable para modificar el comportamiento de la porción distintiva de la entidad de programación distintiva.

[c72] 30. El método de la reivindicación 21, por el cual la información del comportamiento del parche de una entrada de parche seleccionada contiene información utilizable para modificar el comportamiento de la porción distintiva de

la entidad de programación distintiva.

[c73] 31. El método de la reivindicación 21, por el cual la información del comportamiento del parche de una entrada de parche seleccionada contiene una prueba de parámetro de validación a ser aplicado a un parámetro específico asociado con la entidad de programación distintiva.

[c74] 32. El método de la reivindicación 21, por el cual la identidad de las memorias del computador que guardan la estructura de información de la tabla de parches es configurable.

[c75] 33. El método de la reivindicación 21, por el cual las memorias del computador están directamente conectadas a un sistema de computación en el que una o más entradas de parche serán aplicadas.

[c76] 34. El método de la reivindicación 21, por el cual las memorias del computador están directamente conectadas a un sistema de computación en el que no será aplicada ninguna entrada de parche.

[c77] 35. Un sistema de computación que implementa automáticamente los parches de código recibidos, que comprenden:

una biblioteca preinstalada de funciones de ayuda; y

un agente de parcheo que recibe parches de código, cada uno de los cuales tiene como objetivo un grupo de módulos ejecutables y la identificación de una función de ayuda en la biblioteca, y que, cuando se ejecuta un módulo ejecutable en un grupo objetivo de un parche de código recibido, se invoca la función de ayuda identificada con el parche de código que tiene como objetivo el grupo de módulos ejecutables, incluyendo el módulo ejecutable para realizar la validación de valor.

[c78] 36. El sistema de computación de la reivindicación 35, por el cual el agente de parcheo incluye un subsistema de mantenimiento de la biblioteca que recibe nuevas funciones de validación de parámetro, y agrega automáticamente las nuevas funciones de validación de la biblioteca recibidas, de manera que estas funciones nuevas primarias puedan ser invocadas de acuerdo con los parches de código que los identifican.

[c79] 37. Un método de un sistema de computación para aplicar automáticamente a un parche de software, que comprende:

recepción de un parche en el sistema de computación para modificar el comportamiento de una entidad de programación, el parche (1) que especifica una manera para modificar el comportamiento de una entidad de programación, y (2) para cada una de una variedad de versiones de la entidad de programación, (a) la identificación de la versión de la entidad de programación; (b) la especificación de una posición en la versión de la entidad de programación para modificar el comportamiento de la entidad de programación en la manera especificada, y (c) la identificación del código que se espera resida en la posición especificada en la versión de la entidad de programación antes de que el comportamiento de la entidad de programación sea modificado;

determinación automática de una versión distintiva de la entidad de programación entre las versiones de la entidad de programación identificada por el parche esté instalado en el sistema de computación; y

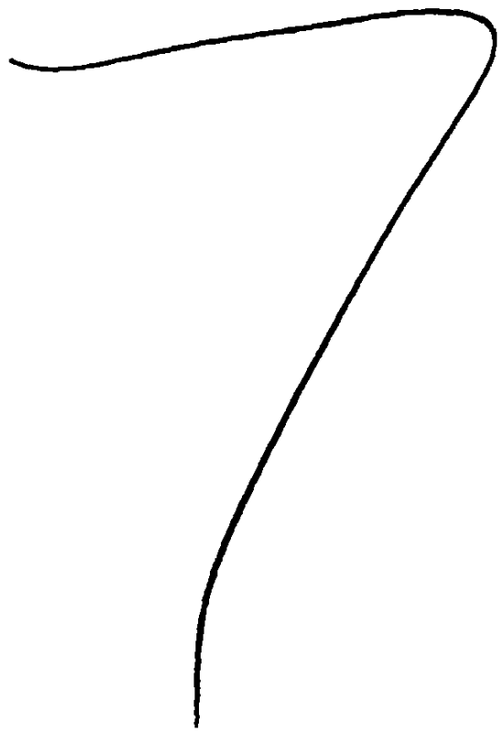
sólo si la posición especificada para la versión distintiva de la entidad de programación contiene el código identificado que se espera resida en la posición especificada, se modifica el comportamiento de la entidad de programación en la posición especificada para la versión distintiva de la entidad de programación de acuerdo con el parche.

[c80] 38. Un medio reconocido por el computador, cuyo contenido lleve a que un sistema de computación realice un método para aplicar automáticamente un parche de software que incluye:

el almacenamiento de un parche en el sistema de computación para modificar el comportamiento de una entidad de programación, el parche (1) especifica la manera en que se debe modificar el comportamiento de la entidad de programación, y (2) para cada una de una variedad de versiones de la entidad de programación, (a) la identificación de la versión de la entidad de programación; (b) la especificación de una posición en la versión de la entidad de programación para modificar el comportamiento de la entidad de programación en la manera especificada, y (c) la identificación del código que se espera resida en la posición especificada en la versión de la entidad de programación antes de que el comportamiento de la entidad de programación sea modificado; determinando

automáticamente de una versión distintiva de la entidad de programación entre las versiones de la entidad de programación identificada por el parche esté instalado en el sistema de computación; y

sólo si la posición especificada para la versión distintiva de la entidad de programación contiene el código identificado que se espera resida en la posición especificada, se modifica el comportamiento de la entidad de programación en la posición especificada para la versión distintiva de la entidad de programación de acuerdo con el parche.



PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE

Compendio de la Revelación 2

Se describe el dispositivo para procesar automáticamente los parches de software. El dispositivo recibe un paquete de parches distintivos en un sistema de computación para modificar el comportamiento de una entidad de programación. El mismo extrae automáticamente del paquete de parches distintivos (1) información de la aplicación del parche que identifica una entidad de programación distintiva a la cual se aplican los parches, y (2) la información del comportamiento del parche que especifica la manera en que debe modificar el comportamiento de la entidad de programación distintiva. El dispositivo agrega automáticamente una entrada distintiva a una tabla de parche que contiene la información extraída de la aplicación del parche, así como información del comportamiento del parche.

TERCERA REIVINDICACIÓN

Reivindicamos:

[c81] 1. Un método en un sistema de computación para aplicar un parche de software utilizando un agente automatizado de parcheo, que incluye:
recepción del parche de software;
como respuesta a la recepción del parche de software sin intervención del usuario:
identificación de una instancia de módulo ejecutable que se encuentra actualmente instalado, al cual corresponde el parche de software recibido; y
la aplicación del parche de software recibido a la instancia del módulo ejecutable que se encuentra actualmente instalado para modificar el comportamiento de la instancia de módulo ejecutable identificada.

[c82] 2. El método de la reivindicación 1, por el cual el módulo ejecutable identificado es un módulo ejecutable de manera independiente.

[c83] 3. El método de la reivindicación 1, por el cual el módulo ejecutable identificado es una biblioteca.

[c84] 4. El método de la reivindicación 1, por el cual el módulo ejecutable identificado es un programa.

[c85] 5. El método de la reivindicación 1, por el cual el módulo ejecutable identificado es un módulo de código administrado.

[c86] 6. El método de la reivindicación 1, por el cual dos parches de software se reciben como correspondientes a la misma instancia de módulo ejecutable, y ambos parches de software recibidos se aplican para modificar el comportamiento de la instancia de módulo ejecutable.

[c87] 7. El método de la reivindicación 6, por el cual dos parches de software recibidos corresponden a ubicaciones diferentes en la instancia del módulo ejecutable.

[c88] 8. El método de la reivindicación 6, por el cual dos parches de software recibidos corresponden a las mismas ubicaciones en la instancia del módulo ejecutable.

[c89] 9. El método de la reivindicación 8, por el cual los parches recibidos se aplican de manera que las dos modificaciones de comportamiento especificadas en los parches, estén exhibidos por la instancia del módulo ejecutable parcheado.

[c90] 10. El método de la reivindicación 8, por el cual los parches recibidos se aplican de manera que sólo la modificación del comportamiento del dominante entre los dos parches de software recibidos, estén exhibidos por la instancia del módulo ejecutable.

[c91] 11. El método de la reivindicación 10, por el cual el parche de software dominante recibido contiene una indicación explícita que es para dominar uno o más parches de software.

[c92] 12. El método de la reivindicación 1, por el cual la instancia de módulo ejecutable identificada exhibe el comportamiento modificado en un momento posterior a la aplicación del parche de software.

[c93] 13. El método de la reivindicación 1, que además incluye, utilizando el agente automatizado de parcheo, la aplicación del parche de software recibido a una imagen de disco para la instancia de módulo ejecutable identificada.

[c94] 14. El método de la reivindicación 1, que además incluye, utilizando el agente automatizado de parcheo, la aplicación del parche de software recibido a la instancia de módulo ejecutable identificada cuando dicha la instancia de módulo ejecutable identificada se reinstale en un momento posterior.

[c95] 15. El método de la reivindicación 1, por el que la instancia de módulo ejecutable actualmente instalada, a la cual corresponde el parche de software, se identifica de entre una variedad de instancias de módulo ejecutables al cual corresponde el parche de software, y por el que el parche de software recibido se aplica utilizando la información específica de instancia de módulo para la instancia de módulo ejecutable identificada incluida en el parche recibido.

[c96] 16. El método de la reivindicación 15, por el cual la variedad de instancias de módulo ejecutables son cada una de éstas una versión diferente del mismo módulo ejecutable.

[c97] 17. El método de la reivindicación 15, por el cual la información del módulo específica a la instancia para el módulo ejecutable identificado contenido en el

parche recibido, especifica una desviación dentro del módulo ejecutable identificado en el cual se modifica el comportamiento del módulo ejecutable identificado.

[c98] 18. El método de la reivindicación 17, por el cual la información del módulo específica a la instancia para el módulo ejecutable identificado contenido en el parche recibido, especifica además el contenido de la instancia de módulo ejecutable identificado que se espera aparezca en una desviación específica antes de aplicar el parche recibido.

[c99] 19. El método de la reivindicación 1, que además incluye, utilizando el agente automatizado de parcheo:

la recepción de un segundo parche de software que corresponde a un módulo ejecutable que no está presente en el sistema de computación; y

como respuesta a la recepción del segundo parche de software, sin la intervención del usuario, el almacenamiento del parche de software recibido para una futura aplicación si una instancia de módulo ejecutable a la que le corresponde el parche de software recibido se instala al sistema más adelante.

[c100] 20. El método de la reivindicación 1, en el que, luego de la aplicación, se desinstala el módulo ejecutable identificado y se reinstala, el método incluye además, la utilización del agente automatizado de parcheo sin la intervención del usuario, reaplicando el parche de software recibido a la instancia del módulo ejecutable identificado para modificar el comportamiento de la instancia del módulo ejecutable identificado reinstalado.

[c101] 21. El método de la reivindicación 1, por el que el parche de software recibido contiene código que especifica cómo modificar el comportamiento de la instancia de módulo ejecutable identificado.

[c102] 22. El método de la reivindicación 1 por el que el parche de software recibido contiene información que especifica cómo modificar el comportamiento de la instancia de módulo ejecutable identificado.

[c103] 23. El método de la reivindicación 1 por el que el parche de software recibido especifica la modificación del comportamiento de la instancia de módulo ejecutable identificado a agregar una prueba y los resultados de la misma que

51

involucran lógica a una porción asignada de la instancia de módulo ejecutable identificado.

[c104] 24. El método de la reivindicación 1 por el que el parche de software recibido especifica la modificación del comportamiento de la instancia de módulo ejecutable identificado a agregar una prueba y los resultados de la misma que involucran lógica a una porción asignada de la instancia de módulo ejecutable identificado.

[c105] 25. El método de la reivindicación 1 por el que el parche de software recibido especifica la modificación del comportamiento de la instancia de módulo ejecutable identificado a agregar una función de validación de parámetro a una función asignada en la instancia de módulo ejecutable identificado, por medio de la identificación de una función de validación de parámetro, que será llamada parte de la ejecución de dicha función asignada.

[c106] 26. El método de la reivindicación 1, por el que el comportamiento modificado de la instancia de módulo ejecutable identificado puede ser activado o desactivado con la utilización de una interfase de configuración de parche.

[c107] 27. El método de la reivindicación 1, por el que la utilización del comportamiento modificado de la instancia de módulo ejecutable identificado puede ser activada o desactivada con la utilización de una interfase de configuración de parche.

[c108] 28. El método de la reivindicación 1, por el que el parche de software se recibe al permitir que dicho parche de software sea copiado a una ubicación predeterminada, y por el que la identificación y la aplicación se realiza automáticamente como respuesta a la copia de dichos parche de software a dicha ubicación predeterminada.

[c109] 29. Un medio reconocido por el computador, cuyo contenido genere que un sistema de computación realice un método para aplicar un parche de software, que incluye, la utilización de un agente automatizado de parcheo:

la recepción del parche de software;

como respuesta a la recepción del parche de software:

la identificación de una instancia de módulo ejecutable correspondiente a dicho

parche de software; y

la aplicación del parche de software recibido a la instancia de módulo ejecutable actualmente en utilización para modificar el comportamiento de dicha instancia de módulo ejecutable.

[c110] 30. El medio reconocido por el computador de la reivindicación 29, por el que dicho medio reconocido por el computador es un medio de almacenamiento de información.

[c111] 31. El medio reconocido por el computador de la reivindicación 29, por el que dicho medio reconocido por el computador es un medio de transmisión de información.

[c112] 32. Un sistema de computación para aplicar el parche de software, que incluye:

un receptor que recibe automáticamente el parche de software;

un subsistema de identificación, que como respuesta recibida por parte del receptor del parche de software, sin la intervención de usuario, identifica una instancia de módulo ejecutable actualmente instalada correspondiente al parche de software recibido; y un subsistema de aplicación de parche, que, como respuesta a la identificación de una instancia de módulo ejecutable actualmente instalada correspondiente a al parche de software recibido por medio del subsistema de identificación, sin la intervención de usuario, aplica el parche de software recibido a dicha instancia de módulo ejecutable actualmente instalada para modificar el comportamiento de dicha instancia de módulo ejecutable identificado.

[c113] 33. Un medio reconocido por el computador que contiene una estructura de información de parche de software que representa un parche de software, por el cual dicha estructura de información de parche de software incluye:

la primera información que identifica uno o más módulos ejecutables de software a los cuales se aplica el parche; y la segunda información que especifica la manera en que dichos módulos ejecutables de software deben ser modificados,

de manera que el contenido de dicha estructura de información de parche de software pueda ser utilizada por un agente automático de parcheo, para

seleccionar un módulo ejecutable de software identificado por la primera información y modificar el módulo ejecutable de software seleccionado en la manera especificada por la segunda información.

[c114] 34. El medio reconocido por el computador de la reivindicación 33, por el que dicho medio reconocido por el computador es un medio de almacenamiento de información.

[c115] 35. El medio reconocido por el computador de la reivindicación 33, por el que dicho medio reconocido por el computador es un medio de transmisión de información.

[c116] 36. El medio reconocido por el computador de la reivindicación 33, por el que la segunda información es un código que especifica la manera en que los módulos ejecutables de software identificados deben ser modificados.

[c117] 37. El medio reconocido por el computador de la reivindicación 33, por el que la segunda información es información que indica la manera en que los módulos ejecutables de software identificados deben ser modificados.

[c118] 38. El medio reconocido por el computador de la reivindicación 33, por el que la estructura de información de parche de software incluye además una firma que demuestra que dicha estructura de información de parche de software fue generada por una autoridad de parche asignada y que no ha sido modificada desde la creación de la firma.

[c119] 39. El medio reconocido por el computador de la reivindicación 38, por el que la identidad de un firmante puede ser distinguido de la firma de la estructura de información de parche de software, y dicha identidad de firmante distinguida puede ser comparada con una identidad de firmante que puede ser distinguida de la firma del módulo ejecutable de software identificado.

PARCHEO EFICIENTE

COMPENDIO DE LA REVELACIÓN 3

Se describe el dispositivo para aplicar los parches de software. Con la utilización de agente de parcheo automático, el dispositivo recibe el parche de software. Como respuesta a la recepción del mismo, el dispositivo, sin la intervención del usuario, realiza lo siguiente: Primero, el dispositivo identifica una instancia de un módulo ejecutable que se encuentra actualmente instalado, y al cual le corresponde el parche de software recibido. Segundo, el dispositivo aplica el parche de software recibido a la instancia de módulo ejecutable identificado que se encuentra actualmente instalado para modificar el comportamiento de dicha instancia de módulo ejecutable identificado.

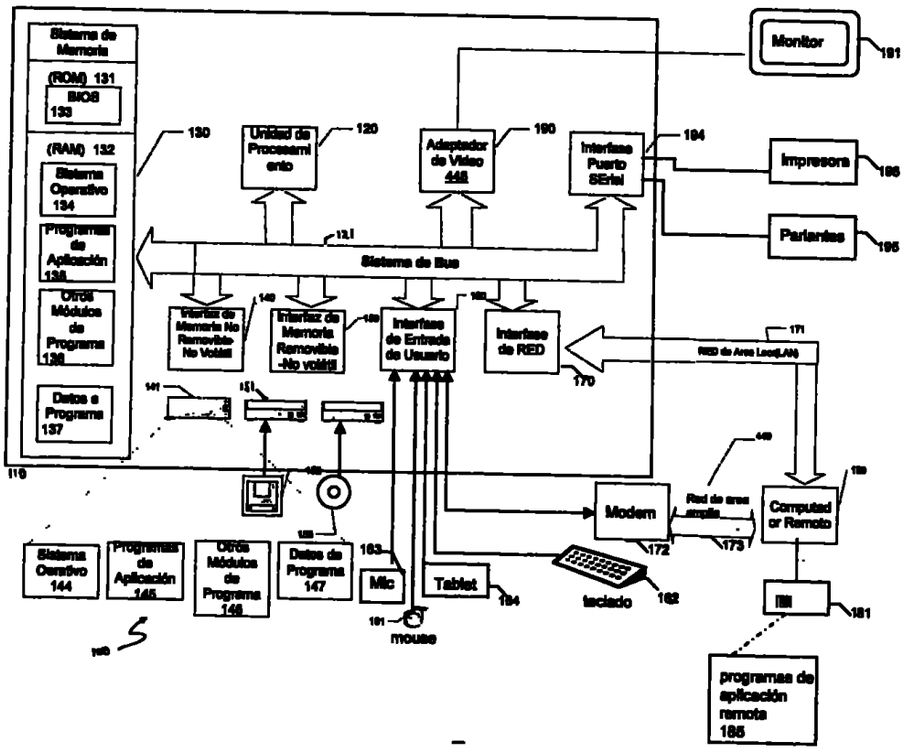

Sistema de Memoria
(ROM) 131
BIOS 133
(RAM) 132
Sistema Operativo 134
Programas de Aplicación 135
Otros Módulos de Programa 136
Datos e Programa 137
110
130
Unidad de Procesamiento
120
Adaptador de Video 448
190
Interfase Puerto SErial
194
Monitor
191
Impresora
196
Parlantes
195
121
Sistema de Bus
Interfaz de Memoria No Removible-No Volátil
140
Interfaz de Memoria Removible -No volátil
150
Interfase de Entrada de Usuario
160
Interfase de RED
170
171
RED de Area Local(LAN)
141
151
Modem
172
Red de area amplia
173
Computador Remoto
180
Sistema Operativo 144
Programas de Aplicación 145
Otros Módulos de Programa 146
Datos de Programa 147
163
Mic
Tablet
194
teclado
162
101
mouse
181
programas de aplicación remota 185

Fig. 1

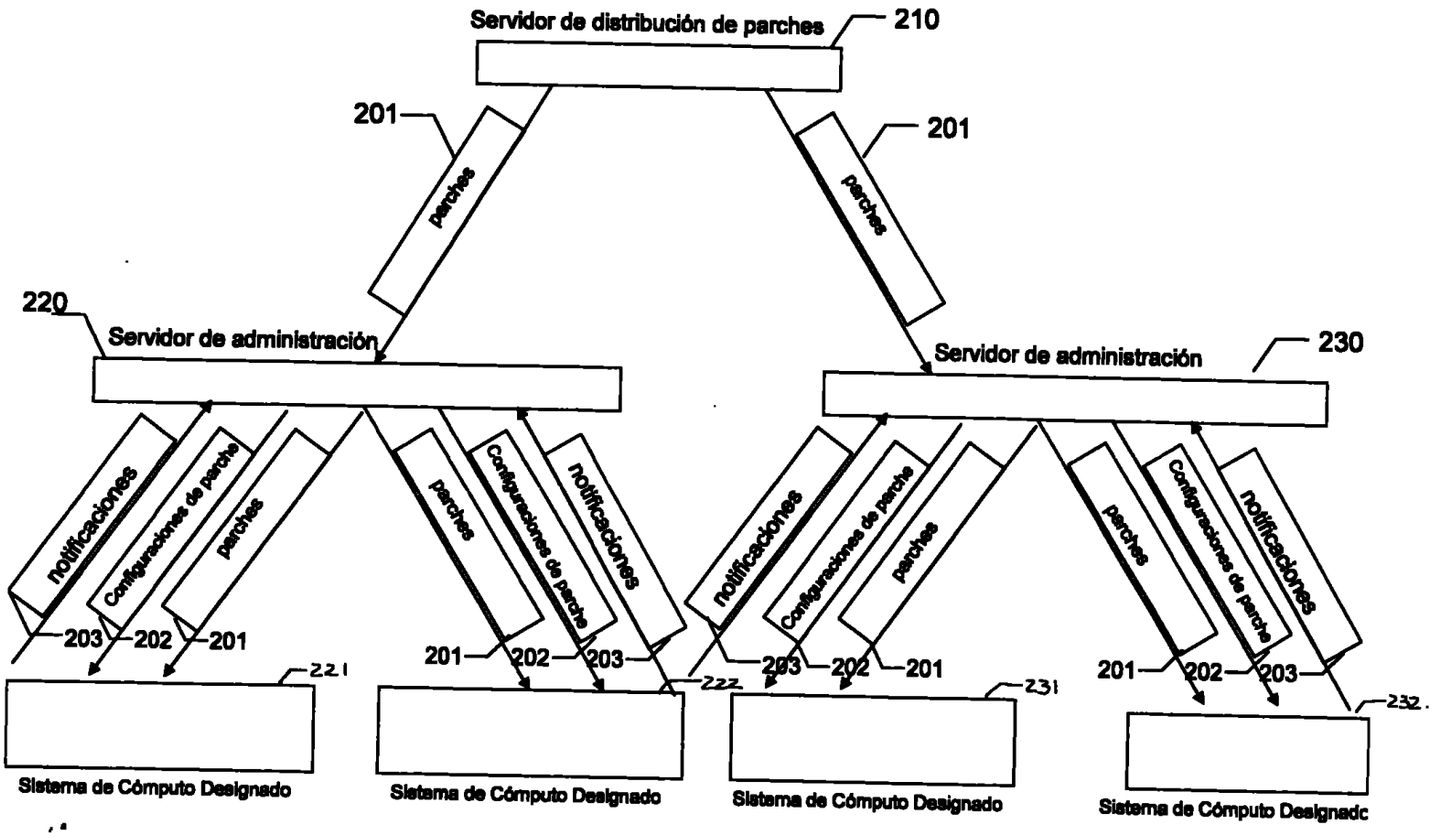

Servidor de distribución de parches
210
201
parches
201
parches
220
Servidor de administración
Servidor de administración
230
notificaciones
Configuraciones de parche
parches
203
202
201
221
Sistema de Cómputo Designado
parches
Configuraciones de parche
notificaciones
201
202
203
222
Sistema de Cómputo Designado
notificaciones
Configuraciones de parche
parches
203
202
201
231
Sistema de Cómputo Designado
parches
Configuraciones de parche
notificaciones
201
202
203
232
Sistema de Cómputo Designadc

Fig. 2

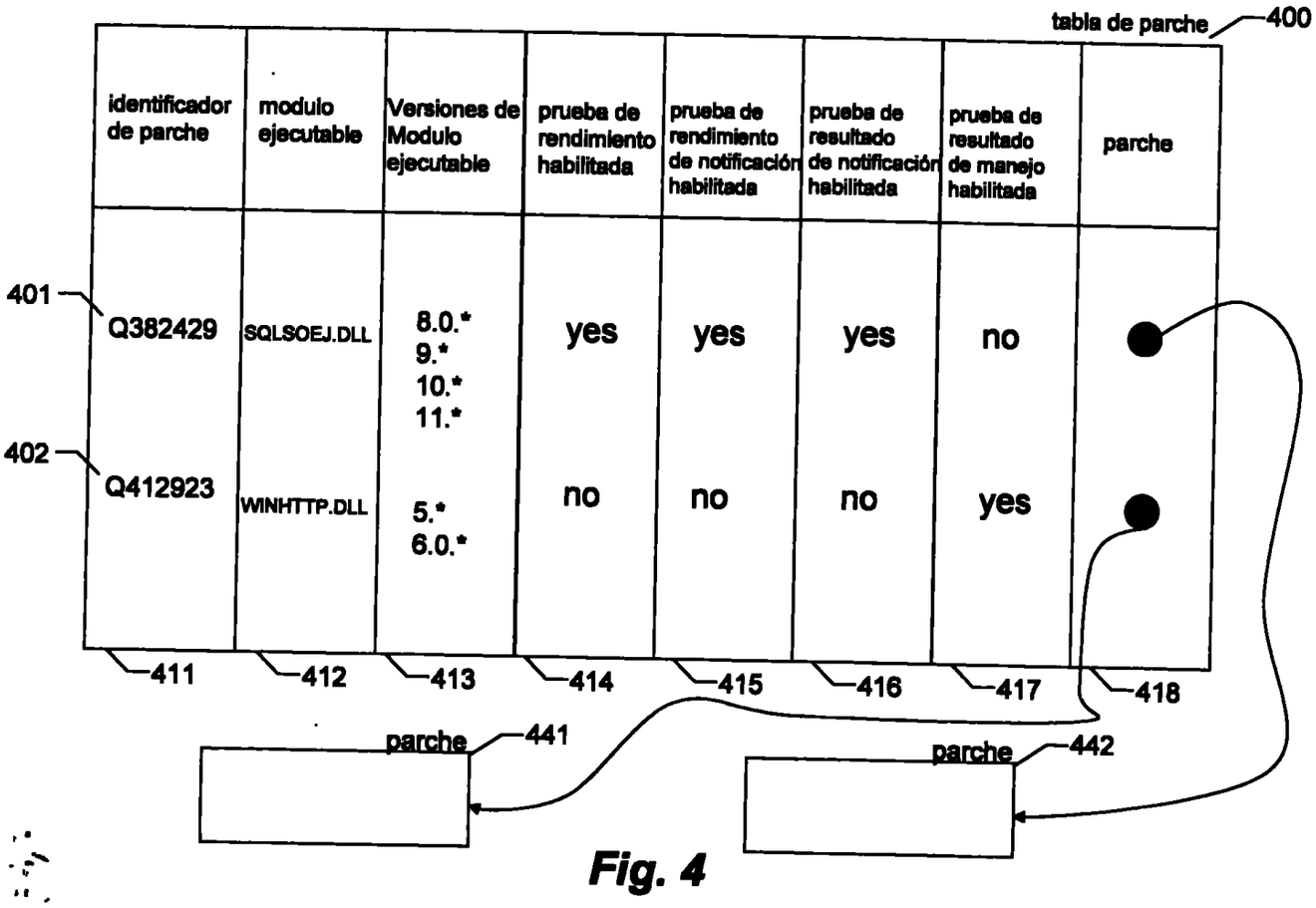


tabla de parche
400
identificador de parche
modulo ejecutable
Versiones de Modulo ejecutable
prueba de rendimiento habilitada
prueba de rendimiento de notificación habilitada
prueba de resultado de notificación habilitada
prueba de resultado de manejo habilitada
parche
401
Q382429
SQLSOEJ.DLL
8.0.*
9.*
10.*
11.*
yes
yes
yes
no
402
Q412923
WINHTTP.DLL
5.*
6.0.*
no
no
no
yes
411
412
413
414
415
416
417
418
parche
441
parche
442

Fig. 4

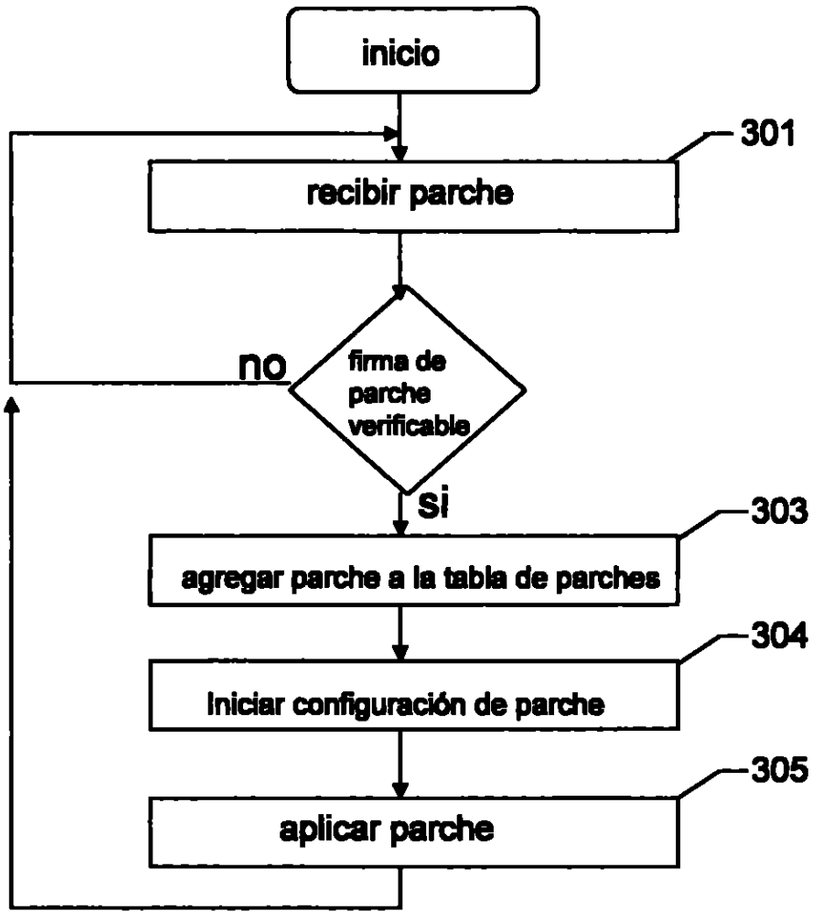

inicio
301
recibir parche
no
firma de parche verificable
Si
303
agregar parche a la tabla de parches
304
Iniciar configuración de parche
305
aplicar parche

Fig. 3

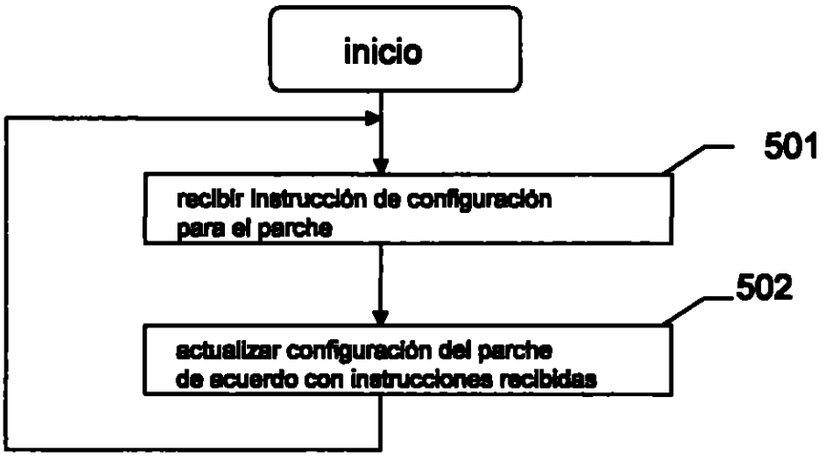

inicio
501
recibir instrucción de configuración para el parche
502
actualizar configuración del parche de acuerdo con instrucciones recibidas

Fig. 5

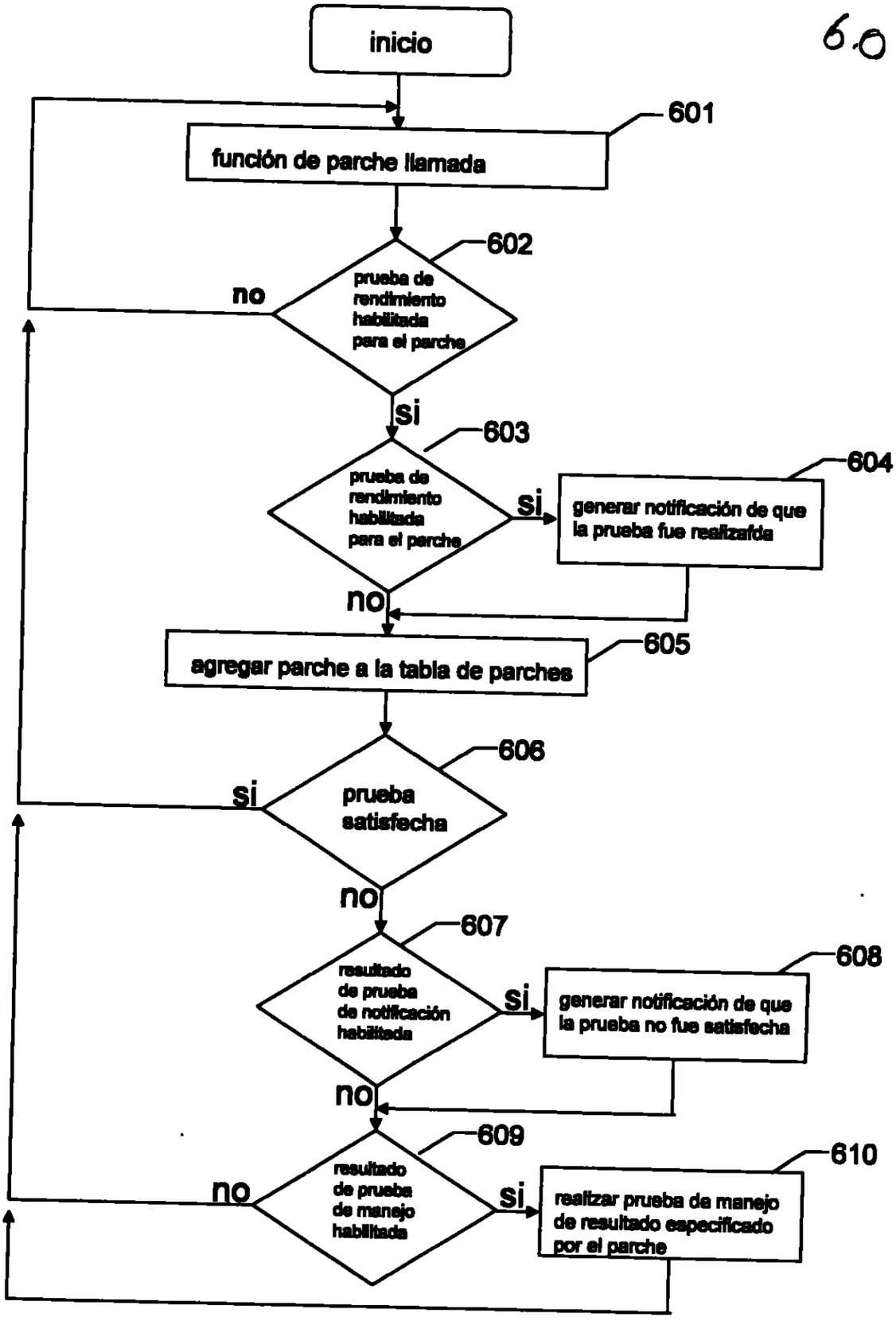

inicio
601
función de parche llamada
602
prueba de rendimiento habilitada para el parche
no
si
603
prueba de rendimiento habilitada para el parche
si
604
generar notificación de que la prueba fue realizafda
no
605
agregar parche a la tabla de parches
606
si
prueba satisfecha
no
607
resultado de prueba de notificación habilitada
si
608
generar notificación de que la prueba no fue satisfecha
no
609
resultado de prueba de manejo habilitada
no
si
610
realizar prueba de manejo de resultado especificado por el parche

Fig. 6

61

TRADUCCIÓN DEL INGLÉS AL ESPAÑ L DE CINCO PÁGINAS DE *PARTE* DE UN DOCUMENTO ORI INAL ENTREGADO EN UN F LDER DE LA FICINA DE PATENTES Y MARCAS REGISTRADAS DEL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS, CON COBERTURA DE PLÁSTICO TRANSPARENTE, REALIZADA POR NORA JIMÉNEZ DE ADAM, TRADUCTORA E INTÉRPRETE OFICIAL. 30 DE MARZO DE 2005

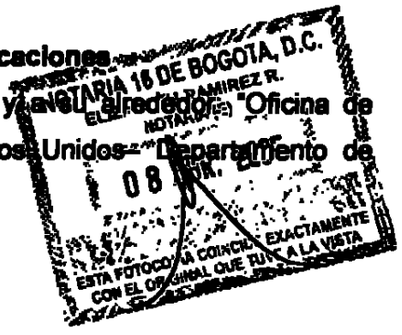
PRIMERA PÁGINA:

Emitida en papelería especial con reborde negro-gris con el número PA1251790 en la parte superior del documento: **"ESTADOS UNIDOS DE AMERICA – PARA TODOS AQUELLOS A QUIENES PUEDA INTERESAR: - DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS – Oficina de Patentes y Marcas Registradas de los Estados Unidos – 24 de noviembre de 2004**

ESTE DOCUMENTO CERTIFICA QUE LOS ANEXOS AL PRESENTE SON COPIA FIEL DE LOS REGISTROS DE LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DE LOS ESTADOS UNIDOS DE AQUELLOS DOCUMENTOS SOBRE LA PETICIÓN PARA PATENTE IDENTIFICADA MÁS ADELANTE QUE CUMPLIÓ CON LOS REQUISITOS PARA QUE SE LE OTORGARA UNA FECHA PARA SU PRESENTACIÓN BAJO 35 USC 111.

N° DE PETICIÓN: ***10/880,709***
FECHA DE PRESENTACIÓN: ***30 de junio de 2004***

Bajo la Potestad del COMISIONADO DE PATENTES Y MARCAS REGISTRADAS
(Firma) E. BORNETT – Funcionario de Certificaciones
Estampilla dorada con emblema en el centro y en el alrededor: "Oficina de Patentes y Marcas Registradas de los Estados Unidos – Departamento de Comercio"



61

PTO/SB/05 (04-04)
Aprobado para uso hasta 07/31/2006/OMB 0651-0032
Oficina de Patentes de los Estados Unidos. DEPARTAMENTO DE COMERCIO
Bajo el Estatuto de Reducción de Papel de 1995, a ninguna persona se le requerirá que responda a una solicitud de información a menos que ésta contenga un número de control válido OMB.

# TRANSMISIÓN DE SOLICITUD DE PATENTE DE UTILIDAD *(Solamente para solicitudes nuevas no provisionales bajo 37 CFR 1.53 (b))*	Numero de Expediente de Abogado	41826.8010US1
	Primer Inventor	ANTHONY BLUMFIELD
	Título	PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE
	Sello de Correo Especial No.	EV336672580US

ELEMENTOS DE SOLICITUD
Ver MPEP Capítulo 600 en relación con el contenido de solicitudes de patentes de utilidad

DIRIGIR A: Comisionado para patentes, P.O. Box 1450, Alexandria, VA. 22313-1450.

1. ☒ Tasa del formato de transmisión (e.g. PTO/SB/17)
2. [ ] Solicitante reclamo estatus de empresa pequeña. Ver 37 CFR 1.27
3. ☒ Especificación — Total Páginas [31]
   - Título descriptivo de la invención
   - Referencia cruzada de las solicitudes relacionadas
   - Declaración en relación con patrocinio Federal R&D
   - Referencia a la lista secuencial, una tabla o un apéndice con listado de computador.
   - Antecedentes de la invención
   - Resumen de la invención.
   - Descripción de los dibujos (si se adjuntan)
   - Descripción detallada
   - Reivindicaciones
   - Extracto
4. ☒ Dibujo(s) (35 U.S.C. 113) — [Total Páginas [5]]
5. Declaración o juramento. — [Total Páginas [2]]
   a. ☒ *Newly executed (original o copia)*
   b. [ ] Copia de una solicitud anterior (37CFR 1.63 (d)) *(para continuación/divisional con encasillado 18 completado)* [ ]
      i. [ ] ELIMINACIÓN DE INVENTOR — Declaración firmada eliminando inventor(es) designado en la solicitud anterior. Ver 37 CFR 1.63 (d)(2) y 1.33 (b).
6. ✕ Hoja de datos de la Solicitud. Ver 37 CFR 1.76
7. [ ] CD-ROM o CDR en duplicado, tabla grande o programa de computador (Apéndice)
8. Presentación de secuencia de nucleótido y/o aminoácido. (En caso de ser aplicable, todo lo necesario)
   a. [ ] Formato Legible en Computador (CRF)
   b. Listado de secuencia de información en:
      i. [ ] CD-ROM o CDR (2 copias) o
      ii. [ ] papel.
   c. [ ] Declaración verificando la identidad de las copias.

PARTES QUE ACOMPAÑAN LA SOLICITUD

9. ☒ Documentos de tenencia (hoja de presentación & documento(s))
10. [ ] 37CFR 3.73(b) Declaración *(cuando existe un cesionario)* [ ] Poder de abogado
11. [ ] Documento de traducción al inglés (si aplica)
12. ☒ Declaración de información presentada. ☒ Copias de Citaciones IDS. (IDS)/PTO-1449
13. [ ] Modificación preliminar
14. ☒ Recibo de retorno de correo (MPEP 503) *(Debe ser específicamente designado)*
15. [ ] Copia del Certificado del documento de prioridad *(en el evento en que se reclame una prioridad extranjera)*
16. [ ] Requerimiento de no publicación bajo el 35 USC 122 (b)(2)(B)(i). El solicitante debe adjuntar la forma PTO/ SB/35 o su equivalente
17. ✕ Otro. CHEQUE

18. Si es una SOLICITUD CONTINUADA marque la casilla apropiada, y proporcione la información de requisitos abajo y en un anexo preliminar o en una Hoja de Datos de Solicitud bajo el 37 CFR 1.76.

[ ] Continuación [ ] Divisional [ ] Continuación-en-parte (CIP) De la solicitud prioritaria No. ________
Información de la solicitud prioritaria: *Examinador* ________ *Unidad Grupo Arte* ________

Para SOLICITUDES CONTINUADAS O DIVISIONALES solamente: La revelación completa de la patente anterior, de la cual el juramento o declaración se presentaron bajo la caja 5 b. anterior, se considera como parte de los documentos revelados de la solicitud continuada o división adjunta y se incorporada a esta por referencia. La incorporación puede solamente ser basada cuando la porción haya sido inadvertidamente omitida de las partes que acompañan la solicitud.

19. DIRECCIÓN DE CORRESPONDENCIA

☒ Número de cliente 25096 o [ ] Dirección de correspondencia abajo

Nombre					
Dirección					
Ciudad		Estado		Código Postal	
País		Teléfono		Fax	
Nombre (en letra de imprenta)	STEVEN D. LAWRENZ		Número de Registro (Apoderado/Agente)	37.376	
Firma	/Ilegible/ Firmado		Fecha	6/29/04	

Copia proporcionada por USPTO de la base de datos de imagen PACR el 11-22-2004

Nota del Traductor: Esta página se encuentra sellada en su parte superior izquierda con Número 15866 U.S. PTO/código de barras No. 063004; y en su parte superior derecha 22553 U.S. PTO 10/880709 código de barras No. 063004.

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 ABR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

63

Transmisión de Gastos para FY 2004
Efectivo 10/01/2003

Los gastos de patentes están sujetos a revisión anual

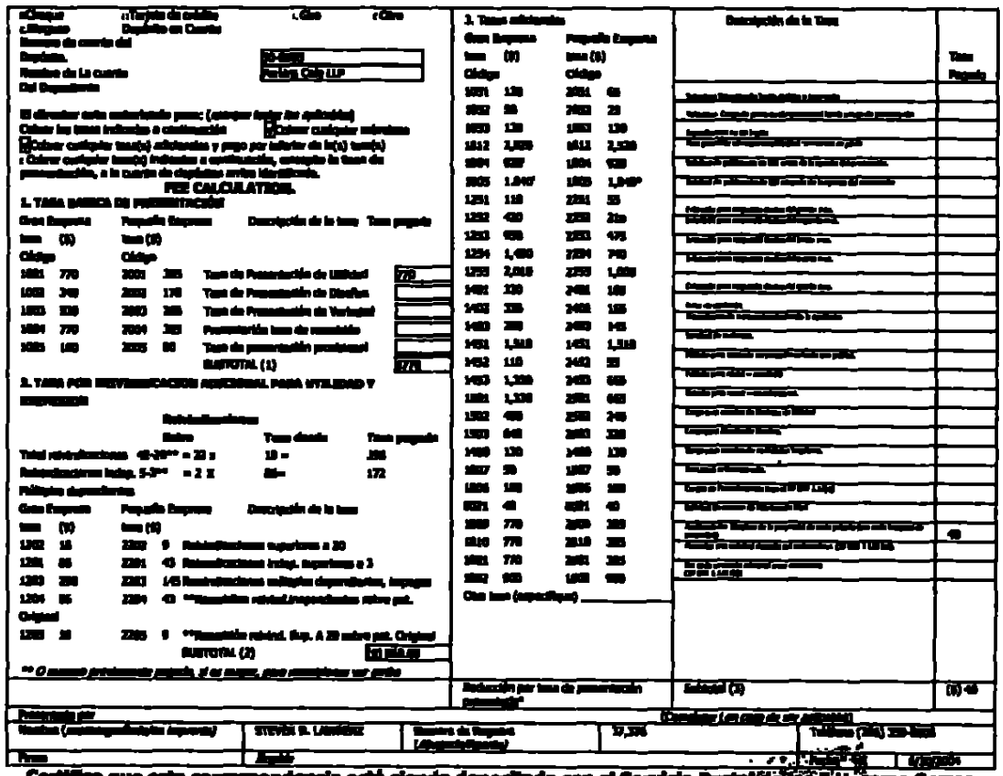
☐ El solicitante reivindica estatus de pequeña empresa. Ver 37CFR 1.27

Valor Total de pago USD$1378.00

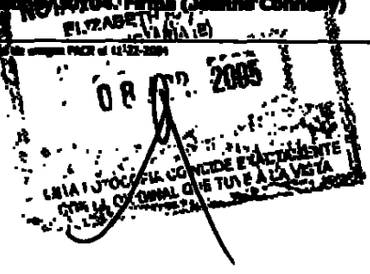
Complete si conoce la información	
Numero de Solicitud	No se ha asignado aún
Fecha de presentación	La presente
Nombre del primer Inventor	Anthony Blumfield
Nombre del Examinador	No ha sido asignado aún
Unidad Inventiva	N/A
Numero de Expediente del Abogado	41826.8010US1

Método de Pago (Revisar todas las aplicables)

Calculo de la tasa (continua)



Certifico que esta correspondencia está siendo depositada con el Servicio Postal de EE.UU. como Correo Expreso N° EV336671580US, en un sobre dirigido a: Commissioner for Patents, P.O. Box 1450, VA 22313-1450 en la fecha que aparece más abajo. Fecha: 08/07/2004. Firma: (Joanna Connelly)



63

CUARTA PÁGINA:

FICHA TÉCNICA DE LA SOLICITUD

Información de la Solicitud

:

Tipo de Solicitud:	**Corriente**
Referencia:	**Utilidad**
Clasificación Sugerida:	
Unidad de Grupo de Arte Sugerida:	**N/A**
¿CD-ROM o CD-R?:	**Ninguno**
¿Presentación de secuencia?:	**Ninguna**
¿Formato legible por el computador? (CRF):	**No**
Título:	**PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE**
Número de Expediente del Abogado:	**41826.8010US1**
¿Solicitud de Publicación Temprana?:	**NO**
¿Solicitud de No Publicación?:	**NO**
Total hojas de Gráficas:	**5**
¿Pequeña Entidad?:	**NO**
¿Petición Incluida?:	**NO**
¿Orden de Confidencialidad en Sol. Principal?	**NO**

FICHA TÉCNICA DEL SOLICITANTE

Tipo de Autoridad del Solicitante:	**Inventor**
País de Ciudadanía Principal:	**Gran Bretaña**
Condición:	**Plena capacidad**
Primer Nombre:	**Anthony**
Apellido:	**Blumfield**
Dirección:	**18229 NE 28th Street, Redmond, WA 98052**

NOTARIA 14 DE BOGOTA, D.C.
ELIZABETH R. ... EZ R.
NOTARIA (E)
9 En 2015
ESTA FOTOCOPIA CONCUERDA CON EL ORIGINAL QUE TUVE A LA VISTA

Tipo de Autoridad del Solicitante:	**Inventor**
País de Ciudadanía Principal:	**Israel**

Condición:	Plena capacidad
Primer Nombre:	Gilad
Apellido:	Golan

Nota al final: Copia suministrada por USPTO de la IFW Image Database el 11/22/2004

QUINTA PÁGINA:

Residencia:	16908 NE 43th. Ct, Redmond, WA 98052
Tipo de Autoridad del Solicitante:	Inventor
País de Ciudadanía Principal:	E.U.
Condición:	Plena capacidad
Primer Nombre:	Jason
Apellido:	Garms
Dirección:	12014 198th Ct. NE Woodinville, WA 98077
Tipo de Autoridad del Solicitante:	Inventor
País de Ciudadanía Principal:	E.U.
Condición:	Plena capacidad
Primer Nombre:	Saud
Apellido:	Alshibani
Dirección:	pmb 152, 700 NW Gilman Blvd. #E-103, Issaquah, WA 98027-5335

NOTARIA 16 DE BOGOTA
ELIZABETH GOMEZ R.
NOTARIA (E)
30 MAR. 2005
25086
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL ORIGINAL QUE RIVE A LA VISTA

Información de Correspondencia

No. de Correspondencia del Cliente:

Información del Cesionario

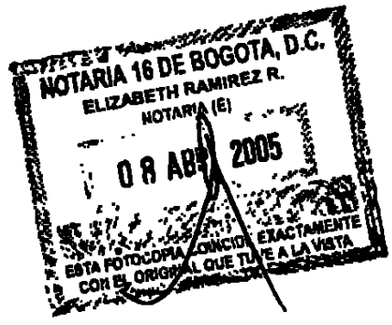
Nombre del Cesionario:	Microsoft Corporation
Dirección de Correspondencia:	One Microsoft Way

Ciudad de Correspondencia:	**Redmond**
Estado o Provincia de la Dirección Postal:	**WA**
Código o Zip de la Dirección Postal:	**98052**

Información de Representación

No. de Representante del Cliente:	**25096**

Nota al final: Copia suministrada por USPTO de la IFW Image Database el 11/22/2004



PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE

67

EL CAMPO TÉCNICO

[0001] La invención presente está dirigida al campo de actualización de la operación de programas de computadora instalados.

EL FONDO

[0002] "Reparchar" o remendar es el proceso de modificar los programas ya-instalados, incluso los programas de aplicación, programas utilitarios, sistemas operativos y componentes del sistema operativo, controladores de dispositivos, etc. Remendar puede ser útil para modificar los programas de una variedad de propósitos, incluyendo la corrección de un error de programación, reduciendo o eliminando un riesgo de seguridad, o mejorando la lógica usada por el programa modificado. El "Reparcheo" es iniciado típicamente por la compañía u otra organización que originalmente proporcionó el programa a ser remendado.

[0003] Los programas instalados están predominantemente compuestos de módulos de código ejecutables. Como ejemplo, muchos programas diseñados para ejecutarse sobre el sistema operativo Windows XP de Microsoft Corp. de Redmond, Washington están predominantemente compuestos de módulos de código ejecutables llamados "DLLs." Un acercamiento convencional popular para remendar es identificar, entre los módulos de código ejecutables que constituyen el programa instalado a ser remendado, el módulo de código ejecutable que contiene el código del programa que uno desea modificar con un parche creando una nueva versión del módulo de código ejecutable identificado en el cual se desea que modificación sea hecha; y distribuyendo la nueva versión del módulo de código ejecutable identificado, junto con un programa instalador a usuarios que pueden desear aplicar el parche. Cada usuario determina entonces si desea aplicar el parche, y en ese caso, ejecuta el programa instalador que

reemplaza la versión original del módulo del código ejecutable identificado con la nueva versión del módulo del código ejecutable identificado.

[0004] Los acercamientos convencionales para remendar tienen varios desventajas significantes. Estas desventajas aumentan a menudo la carga asociada con recibir y aplicar los parches. En algunos casos, este aumento de carga tarda la aplicación de algunos parches por algunos usuarios, e incluso previene la aplicación de algunos parches por algunos usuarios. Tal retraso y prevención en la aplicación de parches pueden en algunos casos tener consecuencias negativas serias para los usuarios, sobre todo para los parches diseñados para reducir o eliminar un riesgo de seguridad.

[0005] Una desventaja de acercamientos convencionales a remendar relacionada a casos comunes en que los parches múltiples deben crearse distribuidos para efectuar una sola modificación a un solo programa. En algunos casos, el programa a ser remendado tiene varios "sabores" diferentes de un módulo de código ejecutable particular, como un sabor diferente para cada sistema operativo o versión del sistema operativo en que el programa esta diseñado para ejecutarse, y/o cada versión del idioma natural del programa. Donde el módulo de código ejecutable identificado es tal un módulo de código ejecutable, la creación del parche y el proceso de distribución descritos para cada sabor del módulo de código ejecutable identificado. El usuario debe seleccionar entonces y debe aplicar el parche para el sabor apropiado del módulo de código ejecutable identificado. El ordenamiento a través del número grande resultante de parches y seleccionar el conjunto apropiado de parches para la aplicación en el sistema de cómputo de cada usuario pueden ser muy pesados, tanto para que esta condición a veces se llame el "infierno del parche". En algunos casos, un administrador debe mantener una base de datos del inventario que identifica el conjunto de versiones del módulo ejecutable instalado en cada sistema designado que es usado para seleccionar los parches convencionales apropiados para cada sistema designado.

[0006] Otra desventaja de acercamientos convencionales para remendar se relaciona al tamaño grande de los parches distribuidos. No es raro para los módulos de código ejecutables tener un tamaño medido en megabytes que a su vez causa que un solo parche tenga ese tamaño comparable, haciéndolo pesado para distribuir y guardar, o incluso imposible de distribuir y guardar, para algunos usuarios. Este problema puede multiplicarse para parches que tienen múltiples sabores. Más allá, porque cada parche convencional incluye típicamente a un suplente completo del módulo ejecutable, mientras se aplican los parches convencionales pueden contribuir al problema de errores de código.

[0007] Una desventaja adicional al acercamiento convencional de remendar se relaciona a una necesidad para algunos usuarios de probar los parches antes de aplicarlos a los sistemas de cómputo de producción. En algunos casos, instalar un parche en un sistema de la computadora puede tener resultados adversos, como dónde la nueva versión del módulo de código ejecutable identificado contenida en el parche introduce un nuevo error de programación, o donde causa una nueva, interacción impredecible con otro programa que corre en el sistema de cómputo contra el cual es aplicado. En concordancia, a menudo antes de aplicar un parche contra un sistema de producción cuyos datos y funcionamiento son importantes de sostener, el usuario aplica el parche primero contra un sistema de prueba para evaluar si el parche es seguro de aplicar al sistema de producción. Tal comprobación separada de parches agrega carga asociada al "reparchamineto". Adicionalmente, dónde un parche convencional crea un problema–como un problema de compatibilidad de aplicación o una nueva vulnerabilidad – en un momento substancial después de que el parche es aplicado, puede ser difícil de rastrear tales problemas detrás del parche.

[0008] Una desventaja adicional de acercamientos convencionales para remendar se relaciona con el funcionamiento del instalador incluido en el parche. A menudo para reemplazar un módulo de código ejecutable que es parte de ejecutar el programa, el instalador debe terminar la ejecución de

09

ese programa primero. Tampoco, en algunos casos, tal reemplazo puede completarse sin reiniciar el sistema de cómputo. Estos dos pasos pueden causar rupturas sustanciales en el uso del sistema de cómputo remendado.

[0009] Otra desventaja de acercamientos convencionales a remendar involucra intentar remendar un módulo ejecutable para el cual "un parche privado", también llamado " parche caliente ó parche urgente – Hot Fix ", ha sido emitido antes para un subconjunto apropiado de clientes para ese módulo ejecutable. En tales casos, debido a dificultades encontradas en la distribución de parches convencionales que sustituyen nuevas versiones diferentes del módulo del código ejecutable que dependiendo de cada usuario que depende si el usuario ha aplicado el parche caliente, en cambio es típico distribuir un parche convencional simple que suple una sola nueva versión del módulo ejecutable independiente de si el usuario ha aplicado el parche caliente. Si esa nueva versión incluye el parche caliente, el parche impone el parche caliente en clientes no pensados para recibirlo. Si, por otro lado, esa nueva versión no incluye el parche caliente, priva a los clientes pensados para recibir el parche caliente del parche caliente.

[0010] Otra desventaja de acercamientos convencionales a remendar involucra el hecho de que los instaladores para productos de software que confían en un módulo ejecutable particular, como una biblioteca de enlace dinámica particular, a menudo "esconden" el módulo ejecutable guardándolo en una ubicación no-normal en el sistema de archivos (filesystem) del sistema de la computadora objetivo. En concordancia, a veces es difícil o imposible determinar si un sistema designado particular contiene una copia del módulo ejecutable a ser remendado, y, en ese caso, dónde reside en el sistema de archivos (filesystem) del sistema de cómputo designado. También, algunos productos de software mantienen un catálogo de versiones de módulo ejecutables que se han instalado por sus instaladores. Un producto de software puede contar con la exactitud de una indicación en el catálogo de versión de un módulo ejecutable particular. Tal confianza es derrotada donde un parche convencional reemplaza la versión

del módulo ejecutable identificada en el catálogo con una nueva versión del módulo ejecutable sin actualizar el catálogo.

[0011] Otra desventaja de acercamientos convencionales para remiendos sueltos desde el hecho que son imposibles de aplicar en un momento antes de que el módulo ejecutable a ser remendado se instala en el sistema de la computadora designada. Como resultado, si el módulo ejecutable a ser remendado se instala en el sistema de la computadora designada después de que un parche convencional para ese módulo ejecutable se recibe, es improbable que el parche se aplique al módulo ejecutable.

[0012] Otra desventaja de acercamientos convencionales para remendar es que ellos pueden aplicarse típicamente sólo por un usuario autenticado en el sistema de la computadora designada que usa una cuenta administrativa que tiene los permisos de modificación libres o habilitados. Autenticarse en una cuenta administrativa para este propósito puede hacer que el sistema de cómputo designado sea vulnerable a virus presentes en el sistema de la computadora designada que busca modificar aspectos del sistema de la computadora designada y permisos libres requeridos para eso.

[0013] Otra desventaja de acercamientos convencionales a remendar es que esos parches convencionales pueden ser difíciles o imposibles de desactivar, requiriendo pasos tales como reversar el reemplazo de un módulo ejecutable, o reversar una o más modificaciones al registro del sistema.

[0014] En concordancia, un nuevo acercamiento a remendar supera algunas o todas las desventajas de acercamientos convencionales para remendar lo discutido anteriormente tendría utilidad significante.

DESCRIPCIÓN BREVE DE LOS DIBUJOS

[0015] La figura 1 ilustra un ejemplo de un ambiente de sistema de informática conveniente en el cual la facilidad puede implementarse.

[0016] La figura 2 es un diagrama de flujo de datos que muestra un intercambio típico de datos entre los sistemas de la computadora de acuerdo con la facilidad.

[0017] La figura 3 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para recibir y procesar un nuevo parche.

[0018] La figura 4 es un diagrama de estructura de datos que muestra una tabla de parche de muestra típico de aquéllos usados por la facilidad.

[0019] La figura 5 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para poner al día las instrucciones de la configuración para un parche particular.

[0020] La figura 6 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para realizar la aprobación del parámetro especificado por un parche.

DESCRIPCIÓN DETALLADA

[0021] Una facilidad del software para remendar el código de programa de computadora instalado ("la facilidad") es proporcionada. En algunas personalizaciones, la facilidad agrega parámetros de prueba y prueban el resultado manejando las funciones instaladas. En otras personalizaciones, la facilidad agrega varios otros tipos de funcionalidades a las funciones instaladas, en algunos casos en posiciones arbitrarias en los flujos de ejecución de las funciones instaladas.

[0022] En algunas personalizaciones, para cada parche, la facilidad distribuye a cada sistema de computadora a ser remendado--es decir, cada "sistema de computadora" designado -- la especificación de un punto en el cual realizar una prueba, la identidad de la prueba a realizar, y cómo actuar en respuesta a uno o más resultados de la prueba diferentes. En algunas personalizaciones, la facilidad proporciona un conjunto normal de validación de parámetro y otras pruebas cuyo uso puede especificarse en los parches. Por ejemplo, un parche puede especificar que, para una función particular,

si un parámetro particular de la función no tiene un cierto valor, la invocación de la función debe fallar antes de que su ejecución substantiva empiece. Otro parche puede especificar que, para una función particular, si un parámetro particular tiene una longitud que excede una longitud máxima especificada, el parámetro debe truncarse a la longitud máxima especificada antes de la ejecución de que la función se permita proceder. Muchos proezas de seguridad se basan en causar que funciones a ser llamadas con valores de parámetro que, mientras ellos no se bloqueen en la versión original del código de una función, causen que la función creer o se aproveche de una condición insegura. En muchos casos, tales ventajas pueden ser prevenidas usando tales parches para prevenir que la función se ejecute con tales valores de parámetro. En algunas personalizaciones, los parches especifican la comprobación de valores de otra manera que los parámetros de la función, tales valores leídos de un archivo o introducidos por el usuario.

[0023] En algunas personalizaciones, un agente remendando automatizado recibe cada parche automáticamente, lo valida, y lo almacena en una tabla de parches para la posible aplicación. En algunas personalizaciones, cada parche se aplica a cualquier instancia del módulo ejecutable a ser remendado que ya está cargado en el sistema de la computadora designada cuando el parche es recibido. Este acercamiento es referido aquí dentro como el "parche caliente – (Hot Fix) ", y permite a los parches ponerse eficazmente inmediatamente al recibirse, y no exige que la facilidad determine donde el módulo ejecutable a ser remendado se guarde en el disco. En algunas personalizaciones, cada parche recibido se aplica a la imagen del disco del módulo ejecutable a ser remendado, para que, cuando la imagen del disco sea cargada un tiempos futuros, la imagen del disco cargada incluya el parche. Este acercamiento referido aquí dentro como el "parche frío – Cold Fix", y permite a los parches ser persistente a través de sesiones múltiples. En algunas personalizaciones, la facilidad realiza el remendando caliente y frío. En algunas personalizaciones, cada parche se

aplica al módulo ejecutable a ser remendado cada vez que el módulo ejecutable a ser remendado es cargado por el cargador del sistema operativo. Este acercamiento es referido aquí dentro como "carga a tiempo del parche – load-time patching." En algunas personalizaciones, cada parche se aplica al módulo ejecutable a ser remendado cada vez que la función a ser remendada es invocada. Este acercamiento es referido aquí dentro como "llamada de interceptación de parche – call-interception patching." Ambas (1), carga a tiempo de parche y llamada a tiempo de parche no exige a la facilidad poder determinar donde el módulo ejecutable a ser remendado se guarda en el disco, (2) la facilidad de reversibilidad lista de un parche particular, y (3) no requiere modificación de la imagen del disco del módulo ejecutable.

[0024] En algunas personalizaciones, la facilidad permite a un usuario o a administrador configurar el funcionamiento de parches que han sido aplicados. Como ejemplos, tal configuración puede incluir, para un parche aplicado en particular: si la prueba especificada por el parche es realizada cuando la ejecución alcanza el punto especificado para el parche; si el resultado de la prueba del manejo especificado por el parche es realizado o ignorado; y/o si el rendimiento de la prueba y/o su resultado es autenticado, desplegado en un mensaje de advertencia, etc. En estas personalizaciones la facilidad permite probar los parches en los sistemas de computadora de producción habilitando y deshabilitando el manejo del resultado autenticado inicialmente. En estas personalizaciones, la facilidad más allá permite la operación del parche a ser autenticado en "un modo verboso" después de habilitar el manejo de su resultado para ayudar con identificar casos en los culaes el parche está creando un problema, como un problema de compatibilidad de aplicación u otro problema IT. Estas personalizaciones también permiten desactivar un parche rápidamente después de ser aplicado si se descubre que el parche está creando problemas. Algunas personalizaciones de la facilidad también permiten que el parche sea

74A

desactivado rápidamente eliminando simplemente el parche del conjunto de parches recibido y guardado en el sistema de computadora designado.

[0025] En algunas personalizaciones, la facilidad usa un "manejo de datos – data-driven" acercamiento de parcheo, en el cual los parches no contienen ningún código, sino datos, como un texto pequeño humano-legible o documento de XML que especifican un punto al cual realizar una prueba, la identidad de la prueba a realizar, y cómo actuar en respuesta a uno o más resultados de prueba diferentes. En tales personalizaciones, un agente de parcheo recibe el manejo de datos (data-driven), y agrega las pruebas y el manejo de la prueba especificados por los parches. En algunas personalizaciones, la facilidad utiliza un acercamiento de parcheo de "manejo de código (code-driven)", y en el cual cada parche contiene un programa corto a ser agregado al módulo ejecutable a ser remendado que realiza la prueba llamando el parámetro normal de la facilidad que prueba las funciones, y que realiza el manejo de la prueba. Usando el parcheo de manejo de datos (data-driven) o el de manejo de código (code-driven), es posible algunas veces direccional todos los sabores de un módulo ejecutable a ser parchado con un solo parche.

[0026] En algunas personalizaciones, la facilidad autentica cada parche para demostrar ambos que (1) el parche es de una fuente aceptado, y (2) no se han modificado los volúmenes del parche desde el tiempo en que el parche se creó por la fuente aceptada.

[0027] En algunas personalizaciones, la facilidad distribuye cada parche para cada sistema de la computadora designada, y un agente de parches en el sistema de la computadora designada determina automáticamente cuales parches para ser aplicados en el sistema de la computadora designada, y cómo serán aplicados, basado en las características del sistema de la computadora designada. Esto releva a los usuarios y administradores de muchas de las cargas convencionalmente asociadas con seleccionar y aplicar los parches, y la carga de mantener una base de datos del inventario exacto y actual. Por ejemplo, estas características

pueden incluir qué versión del módulo ejecutable a ser remendado se instala en el sistema de la computadora designada. En estas personalizaciones, la facilidad puede superar el tipo de problemas causado por los parches calientes, distribuyendo parches que especifican el manejo diferente para los parches caliente y no calientes (hot-fixed y un-hot-fixed) de los diferentes sabores del módulo ejecutable en particular, obviando típicamente alguna necesidad de cualquier sacrificio de un parche caliente para un módulo ejecutable particular o hacer que el parche caliente sea ubicuo al remendar ese módulo ejecutable.

[0028] En algunas personalizaciones, el agente de parches almacena cada parche recibido en el sistema designado, independiente de si el módulo ejecutable a ser remendado por un parche particular se instala en el sistema designado cuando ese parche se recibe. Porque la facilidad en muchos casos aplica los parches en respuesta a la carga de un módulo ejecutable a ser remendado o la invocación de una función a ser remendada, la facilidad puede aplicar un parche a un módulo ejecutable que se instaló en el sistema designado después de que ese parche se recibió en el sistema designado. También, los parches pueden sobrevivir la desinstalación y la reinstalación subsiguiente del módulo ejecutable a ser remendado.

[0029] En algunas personalizaciones, el agente de parches es implementado en un servicio del sistema operativo. En estas personalizaciones, la facilidad otorga al agente de parches cualquier permiso exigido para aplicar un parche. Estas personalizaciones reducen el riesgo de seguridad impuesto típicamente cuando los parches convencionales son aplicados, cuando ellos obvian alguna necesidad para un usuario de autenticarse en el sistema de la computadora designada que usa una cuenta administrativa que tiene los permisos de modificación habilitados, y por eso dando a cualquier virus presente en el sistema de la computadora designada una oportunidad mayor de modificar aspectos sensibles del sistema de la computadora designada.

77

[0030] Los parches usados por la facilidad son típicamente relativamente pequeños, y por consiguiente imponen requisitos de recursos modestos para la transmisión y almacenamiento. También, porque los parches usados por la facilidad tienden a modificar el comportamiento del software remendado en un número pequeño de maneras bien-definidas, la facilidad ayuda a reducir el problema de código basura.

[0031] La figura 1 ilustra un ejemplo de un sistema de cómputo conveniente ambiente 100 en el cual la facilidad puede llevarse a cabo. El sistema de cómputo ambiente 100 es sólo un ejemplo de un ambiente de cómputo conveniente y no se piensa en cualquier limitación acerca del alcance de uso o funcionalidad de la facilidad. Tampoco debería ser interpretado el ambiente de cómputo 100 como si tuviera cualquier dependencia o requisito relacionado a cualquier combinación de componentes ilustrados en el ambiente operativo 100 ejemplar.

[0032] La facilidad es operacional con numerosos otros propósitos de propósito general o especial de los ambientes de cómputo del sistema o configuraciones. Los ejemplos de sistemas de cómputo bien conocidos, ambientes, y/o configuraciones que pueden ser convenientes para el uso con la facilidad incluyen, pero no se limita a: las computadoras personales, los servidores, los dispositivos portátiles o portables, los dispositivos de tabla (tablet devices), los sistemas multiprocesador, los sistemas basados en microprocesador, las cajas de juego, electrónicosa de consumo programables, red PCs, los miniordenadores, los sistemas informáticos grandes, ambientes de cómputo distribuidos que incluyen cualquiera de los sistemas anteriores o dispositivos, y similares.

[0033] La facilidad puede describirse en el contexto general de instrucciones de computadora-ejecutables, como los módulos de programa, ejecutándose por una computadora. Generalmente los módulos de programa incluyen las rutinas, programas, objetos, componentes, estructuras de datos, y similares, qué realizan tareas particulares o implementan tipos de datos

77

78

abstractos particulares. La facilidad también puede practicarse en ambientes de la informática distribuídos dónde las tareas son realizadas por dispositivos del proceso remotos que se unen a través de una red de comunicaciones. En un ambiente de la informática distribuído, pueden localizarse los módulos del programa en los medios de comunicación de almacenamiento de computadora locales y/o remotos incluso los dispositivos de almacenamiento de memoria.

[0034] Con referencia a la Figurar 1, un sistema ejemplar para implementar la facilidad incluye un dispositivo de cómputo de propósito general en la forma de una computadora 110. Los componentes de la computadora 110 pueden incluir, pero no se limita a, una unidad de proceso 120, un sistema de memoria 130, y un sistema de bus 121 que se acopla a varios componentes del sistema incluso la memoria del sistema a la unidad de proceso 120. El sistema de bujs 121 puede ser cualquiera de varios tipos de estructuras de bus incluso un bus de memoria o controlador de memoria, un bus periférico, y un bus local que usan cualquiera de una variedad de arquitecturas de bus. Por vía del ejemplo, y sin limitación, tales arquitecturas incluyen el bus de la Industria de Arquitectura Estandar (ISA), el bus de Arquitectura de Micro Canal (MCA), el bus ISA extendido (EISA), el bus local de la Asociación de Estándares de Video Electrónico (VESA) el bus local, y el Componente de Interconección Periférico (PCI) también conocido como el bus del entresuelo.

[0035] La computadora 110 incluye una variedad de medios de comunicación típicamente legibles por computadora. Los medios de comunicación legibles por computadora pueden ser cualquier medios de comunicación disponible que puede accederse por la computadora 110 y puede ser incluidos ambos medios de comunicación volátil y no volátil, y los medios de comunicación removibles y no removibles. Por vía del ejemplo, y sin limitación, los medios de comunicación legibles por computadora pueden comprender medios de comunicación de almacenamiento de computadora y medios de comunicación. Los medios de comunicación de

76

79

almacenamiento de computadora incluyen volátil y no volátil, los medios de comunicación removibles y no removibles implementados en cualquier método o tecnología para el almacenamiento de información como las instrucciones legibles por computadora, las estructuras de datos, los módulos del programa u otros datos. Los medios de comunicación de almacenamiento de computadora incluyen, pero no se limita a, RAM, ROM, EEPROM, memoria de destello (flash memory) u otra tecnología de memoria, CD-ROM, los discos versátiles digitales (DVD) u otro almacenamiento de disco óptico, casetes magnéticos, cinta magnetofónica, almacenamiento de disco magnético u otros dispositivos de almacenamiento magnéticos, o cualquier otro medio que pueda usarse para guardar la información deseada y qué pueda ser accedida por la computadora 110. Los medios de comunicación incluyen las instrucciones legibles típicamente por computadora, las estructuras de datos, módulos de programa u otros datos en señales de datos moduladas como una onda portadora u otro mecanismo de transporte e incluyen cualquier medio de comunicación de entrega de información. El término "señal de datos modulada" significa que tiene uno o más de su conjunto de características o cambios de tal manera que codifica la información de la señal. Por vía del ejemplo, y sin limitación, los medios de comunicación incluyen los medios de comunicación alambrados como una red alambrada o conexión alambrada directa, y los medios de comunicación inalámbricos como el acústico, RF, infrarrojo y otros medios de comunicación inalámbricos. Las combinaciones de cualquiera de los anteriores también debe ser incluido dentro del alcance de medios de comunicación legibles por computadora.

[0036] El sistema de memoria 130 incluye los medios de comunicación de almacenamiento de computadora en la forma de memoria volátil y no volátil y/o tal como la memoria de solo lectura (ROM) 131 y la memoria de acceso aleatorio (RAM) 132. Un sistema básico de entrada/salida 133 (BIOS), conteniendo las rutinas básicas que ayudan a transferir la información entre los elementos dentro de la computadora 110, tal como durante el inicio, que

NOTARIA 16 DE BOGOTA D.C.
ELIZABETH RAMIREZ (E)
APR 2005
ESTA FOTOCOPIA CONCUERDA EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

79

se guarda típicamente en la ROM 131. La RAM 132 típicamente contiene los datos y/o módulos del programa que son inmediatamente accesibles y/o presentemente siendo operados por la unidad de proceso 120. Por vía de ejemplo, y sin limitación, la figure 1 ilustra el sistema operativo 134, la aplicación del programa 135, otros módulos de programa 136 y programas de datos 137.

[0037] La computadora 110 también puede incluir otros medios de comunicación de computadora removibles y no removibles, volátiles y no volátiles. Por vía del ejemplo únicamente, la figure 1 ilustra un controlador de disco duro 141 que lee desde o escribe a un no removible, no volátil de los medios de comunicación magnéticos, una unidad de disco 151 magnética lee desde o escribe a un removible, no volátil disco 152 magnético, y una unidad de disco 155 óptica lee desde o escribe a un trasladable, no volátil disco 156 óptico como un CD ROM u otros medios de comunicación ópticos. Otro removible/no removible, volátil/no volátil medio de comunicación de almacenamiento de computadora que pueden usarse en el ambiente ejemplar que se opera incluyen, pero no se limita a, los casetes de cinta magnetofónica, tarjetas de memoria de destello, discos versátiles digitales, cinta de video digital, RAM de estado sólida, ROM de estado sólido, y similares. La unidad de disco duro 141 se conecta típicamente al sistema de bus 121 a través de una memoria non-removible como la interfaz 140, y la unidad de disco magnética 151 y la unidad de disco óptico 155 son típicamente conectados al sistema de bus 121 por una interfase de memoria removible 150.

[0038] Los controladores y sus medios de comunicación de almacenamiento de computadora asociados, discutidos anteriormente e ilustrados en la figura 1, proporcionan almacenamiento de instrucciones legibles por computadora, las estructuras de datos, los módulos de programa y otros datos para la computadora 110 en la figura 1, por ejemplo, el controlador del disco duro 141 es ilustrado como sistema operativo de almacenamiento 144, los programas de aplicación 145, otros módulos de programa 146 y

programas de datos 147. Note igual que estos componentes pueden ser diferentes o los mismos del sistema operativo 134, el programa de aplicación 135, otros módulos de programa 136, y datos de programa 137. El sistema operativo 144, el programa de aplicación 145, otros módulos de programa 146, y datos de programa 147 a los cuales son dados aquí dentro números diferentes para ilustrar que, en un mínimo, ellos son copias diferentes. Un usuario puede entrar las órdenes e información en la computadora 110 a través de los dispositivos de entrada como tabla, o digitalizador electrónico, 164, un micrófono 163, un teclado 162 y un dispositivo de apuntamiento 161, normalmente llamado ratón, trackball o almohadilla de tacto. Otros dispositivos de entrada no mostrados en la figura 1 pueden incluir una palanca de mando, la almohadilla de juego, plato satélite, escáner, o similares. Se conectan a menudo éstos y otros dispositivos de entrada a la unidad de proceso 120 a través de una interfaz de entrada de usuario 160 que se acopla al bus del sistema, pero puede conectarse por otra interfaz y estructuras de bus, como un puerto paralelo, un puerto de juego o un bus de serie universal (USB). Un monitor 191 u otro tipo de dispositivo de despliegue también se conecta al sistema de bus 121 vía una interfaz, como una interfaz de video 190. El monitor 191 también puede integrarse con una pantalla de tacto o similares. Note que el monitor y/o el tablero de pantalla de tacto puede acoplarse físicamente a un alojamiento en el cual el dispositivo de cómputo 110 está incorporado, como en un computador personal tipo tabla. Además, las computadoras como el dispositivo de cómputo 110 también puede incluir otros dispositivos de rendimiento periféricos como parlantes 195 e impresora 196 que pueden conectarse a través de una interfaz periférica de salida 194 o similares.

[0039] La computadora 110 puede operar en un ambiente conectado a una red de computadoras que use conexiones lógicas o a las computadoras más remotas, como una computadora remota 180. La computadora 180 remota puede ser una computadora personal, un servidor, un enrutador, una red de PCs, un dispositivo par u otro nodo de la red común, y

típicamente incluye muchos o todos los elementos descritos arriba relacionados a la computadora 110, aunque sólo un dispositivo de almacenamiento de memoria 181 se ha ilustrado en la figura 1. Las conexiones lógicas pintadas en la figura 1 incluyen una red de área local (LAN) 171 y una red de área amplia (WAN) 173, pero también puede incluir otras redes. Tales ambientes de gestión de redes son comúnes en las oficinas, en las redes de computadores de empresas grandes, intranets e Internet. Por ejemplo, en la facilidad presente, el sistema de cómputo 110 puede comprender la fuente de máquina de la cual los datos están siendo migrados, y la computadora remota 180 puede comprender la máquina del destino. Note sin embargo que esas máquinas fuente y destino no necesita ser conectados por una red o cualquier otro medio, pero en cambio, pueden emigrarse los datos por cualquier medio de comunicación capaz de ser escrito por la plataforma de fuente y pueden leerse por la plataforma o plataformas destino.

[0040] Cuando es usado en un ambiente de red LAN que conecta una red de computadoras, la computadora 110 se conecta a la LAN 171 a través de una interfaz de red o adaptador 170. Cuando es usado en un ambiente de redes WAN, la computadora 110 incluye un módem 172 u otro medio típicamente para establecer las comunicaciones sobre la WAN 173, como Internet. El módem 172 puede ser interno o externo y puede conectarse al sistema de bus 121 vía la interfaz de usuario 160 u otro mecanismo apropiado. En un ambiente de red conectado, los módulos de programa graficados relacionados a la computadora 110, o parte de ello, puede guardarse en el dispositivo de almacenamiento de memoria remoto. Por vía de ejemplo, y sin limitación, la figure 1 ilustra el programa de aplicación remota 185 como residente en el dispositivo de memoria 181. Se apreciará que las conexiones de red mostradas son medios ejemplares y otros medios para establecer un eslabón de comunicaciones entre las computadoras pueden usarse.

[0041] Mientras varias funcionalidades y datos son mostradas en la figura 1 como residentes en sistemas de cómputo particulares que son colocados de una manera particular, estos experimentados en el arte apreciarán que tales funcionalidades y datos pueden distribuirse de varias otras maneras por los sistemas de cómputo en los diferentes arreglos. Mientras los sistemas de cómputo configurados como los descritos anteriormente son típicamente usados para apoyar el funcionamiento de la facilidad, una habilidad ordinaria en el arte apreciará que la facilidad puede ser implementada usando dispositivos de varios tipos y configuraciones, y teniendo varios componentes.

[0042] La figura 2 es un diagrama de flujo de datos que muestra un intercambio típico de datos entre los sistemas de cómputo de acuerdo con la facilidad. Los sistemas de cómputo mostrados en la figura 2 (sistemas de cómputo 210, 220, 221, 222, 230, 231, y 232) típicamente tienen algunos o todos los componentes mostrados y discutidos junto con la figura 1. En el servidor de distribución de parches, la facilidad genera uno o más parches. Estos parches 201 se envían del servidor de distribución de parche a uno o más servidores de administración, como los servidores de administración 220 y 230. Cada servidor de administración, a su vez, adelanta los parches a uno o más sistemas de cómputo designados, como la computadora designada sistemas 221, 222, 231, y 232. En algunas personalizaciones (no mostradas), el servidor de distribución de parches envía los parches directamente a uno o más sistemas de cómputo designados, vía una ruta más indirecta que a través de un solo servidor de administración. Se procesan los parches recibidos en un sistema de cómputo designado como descrito en mayor detalle debajo. Un servidor de administración también puede enviar comandos de la configuración del parche 202 a uno o más sistemas de cómputo designados que aplican los comandos de configuración del parche que reconfiguran la operación de parches particulares. Como se discute en mayor detalle debajo, un parche puede desactivarse completamente; si un parche no es inválido, la notificación de

su funcionamiento y su manejo de resultado de prueba puede habilitarse o deshabilitarse a cada uno independientemente. Cuando la notificación de su funcionamiento se habilita, pueden desplegarse las notificaciones o pueden guardarse localmente en el sistema de la computadora designada, o pueden enviarse como notificaciones 203 a un servidor de administración apropiado.

[0043] La figura 3 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para recibir y procesar un nuevo parche. En el paso 301, la facilidad recibe un parche. El parche recibido en el paso 301 puede ser un parche de manejo de datos o un parche de manejo de código. Una muestra de parche de código de datos se muestra en la tabla 1 debajo.

```
1    <Softpatch Patch="Q382429">
2     <AffectedApplication AffectedExe="sqlservr.exe">
3      <AffectedVersion Version="9.*">
4       <AffectedModules Name="SQLSORT.DLL">
5        <Version "8.0.*, 9.*">
6              <Function Name="SsrpEnumCore" Address="0x0802E76B"
7                  Param="2" Paramtype="LPSTR">
8               <Filter MaxByteLength="50" />
9               <Resolution ActionType="BOOL" Action="FALSE" />
10             </Function>
11       </Version>
12       <Version "10.*, 11.*">
13             <Function Name="SsrpEnumCore" Address="0x0802D283"
14                 Param="2" Paramtype="LPSTR">
15              <Filter MaxByteLength="128" />
16              <Resolution ActionType="BOOL" Action="FALSE" />
17             </Function>
18       </Version>
19      </AffectedModules>
20     </AffectedVersion>
21    </AffectedApplication>
22
23    <Signature Hash="MD5" Signature="C509-04AA-9101-8C52-
24     9F6D-BF5A-AEF2-ECE1-0038-34D1"/>
25   </Softpatch>
```


84

TABLA 1

[0044] La línea 1 contiene un único identificador para el parche. La línea 2 identifica la aplicación afectada por el parche. La línea 3 identifica la versión de la aplicación afectada por el parche. La línea 4 identifica el módulo ejecutable afectado por el parche. La línea 5 identifica dos versiones del módulo ejecutable afectado–versiones 8.0. * y 9. *– para los cuales las direcciones del parche son proporcionados. Las líneas 6-10 contienen las direcciones de parche para estas dos versiones del módulo ejecutable. Las líneas 6-7 identifican la función a ser remendada, su dirección en el módulo ejecutable, su parámetro a ser probado por el parche, y el tipo del parámetro a ser probado. La línea 8 indica que el parámetro identificado en las líneas 6-7 debe probarse para determinar si su longitud excede 60 bytes. La línea 9 indica que la invocación de la función debe fallar si la prueba tiene éxito. La línea 12 identifica dos ó más versiones del módulo ejecutable afectado – las versiones 10. * y 11. *– para las cuales las direcciones del parche se proporcionan. Las líneas 13-17 contienen las direcciones del parche para estas dos versiones del módulo ejecutable. Las líneas 13-14 identifican la función a ser remendada, su dirección en el módulo ejecutable, su parámetro a ser probado por el parche, y el tipo del parámetro a ser probado. Puede verse que la dirección de la función a ser remendada en el módulo ejecutable identificado en las líneas 13-14 para versiones 10. * y 11. * es diferente de la dirección de la función a ser remendada identificada en las líneas 6-7 para las versiones 8.0. * y 9. *. La línea 15 indica que el parámetro identificado en las líneas 13-14 que debe probarse para determinar si su longitud excede 128 bytes. La línea 16 indica que la invocación de la función debe fallar si la prueba tiene éxito. El parche puede especificar una variedad de tipos de acción de manejo de resultado, incluso el fracaso de la invocación de la función remendada, levantando una excepción, terminando el proceso en que el módulo ejecutable remendado está ejecutándose o corrigiendo el valor

erróneo (tal como truncando una cadena demasiado larga). Las líneas 23-25 contienen una firma para el parche que identifica la fuente del parche y verifica que el parche no ha sido alterado desde de su fuente.

[0045] La Tabla 2 debajo contiene una versión de manejo de código (code-driven) del parche mostrado anteriormente en la Tabla 1.

```
1     00411A7E  push      3Ch
2     00411A80  mov       eax,dword ptr [str]
3     00411A83  push      eax
4     00411A84  call      ValidateStringLength (411082h)
5     00411A89  add       esp,8
6     00411A8C  movzx     ecx,al
7     00411A8F  test      ecx,ecx
8     00411A91  je        411A9Ah
9     00411A93  jmp       foo+2 (411AD2h)
10    00411A9A  xor       eax,eax
11    00411A9C  ret
```

TABLA 2

[0046] Las líneas 1-3 empujan los parámetros para la función de prueba hacia la pila. La línea 4 llama la función de prueba. Las líneas 5-8 dependen del código de retorno para la función de comprobación. Si la función de prueba tuviera éxito, la línea 9 retorna para empezar ejecutando el cuerpo de la función remendada. Si la función de prueba fallara, las líneas 10-11 empujan un código de resultado de fracaso hacia la pila y retorna desde la función de reparcheo al color de la función remendada. Para la legibilidad, la Tabla 2 omite ciertos detalles presentes en algunos parches de manejo de código (code-driven), incluso una firma comprobable, las instrucciones que prueban para los valores actuales de las banderas de configuración del parche, y las instrucciones relocalizadas desde el principio del código para la función de reparcheo.

[0047] En algunas personalizaciones, los parches de ambos tipos pueden contener la información adicional incluyendo para cada uno de una o más versiones del módulo ejecutable para ser parchado, una firma de archivo que puede usarse para validar que un caso particular del módulo ejecutable es una copia apropiada de esa versión. Tal archivo de firma puede ser, por

ejemplo, un tamaño o una verificación para la versión del módulo ejecutable completo, o el código esperado a ocurrir en un punto particular en el módulo ejecutable, como el desplazamiento al que el módulo ejecutable será remendado.

[0048] En el paso 302, si el parche se firma con una firma válida, entonces la facilidad continúa al paso 303, el resto de la facilidad continúa al paso 301 para recibir el próximo parche. En el paso 303, la facilidad agrega el parche a una tabla del parche local. En el paso 304, la facilidad inicializa la configuración inicial del parche, como enviándolo a una configuración predefinida.

[0049] La figura 4 es un diagrama de estructura de datos que muestra una tabla de parche de muestra típico de aquéllos usados por la facilidad. La tabla del parche 400 contiene las filas, como las filas 401 y 402, cada una dividida en las columnas siguientes: una columna identificadora del parche 411, conteniendo el identificador del parche extraído del parche,; una columna del módulo ejecutable 412, conteniendo la información que identifica el módulo ejecutable a ser reparchado, tal como su nombre,; una columna de versiones de módulo ejecutables 413 que identifica todas las versiones del módulo ejecutable identificado en la columna 412 al que el parche aplica; una columna habilitada de rendimiento de prueba 414, conteniendo el valor de la configuración actual cual por si la prueba especificada por el parche debería realizarse cada vez que la función de reparchado es llamada; una columna habilitada de notificación de rendimiento de prueba 415, conteniendo el valor de la configuración actual por si una notificación debe generarse cada vez que la prueba del parche se ha realizado; una columna habilitada de notificación de resultado de prueba 416, conteniendo el valor de la configuración actual por si una notificación debería generarse cada vez que la prueba del parche tiene éxito; una columna habilitada de manejo de resultado de la prueba 417, conteniendo el valor de la configuración actual por si el manejo de resultado del parche debe llevarse a cabo cuando fallan las pruebas del parche; y una

[0052] La figura 6 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para realizar la aprobación del parámetro especificado por un parche. En el paso 601, una función de parcheo es llamada. En el paso 602, si la prueba es habilitada para el parche que afecta la función llamada, entonces la facilidad continúa en el paso 603, sino, la facilidad continúa en el paso 601 para procesar la próxima llamada a una función de parcheo. En el paso 603, si la notificación de rendimiento de prueba se habilita para el parche, entonces la facilidad continúa en el paso 604, sino la facilidad continúa en el paso 605. En el paso 604, la facilidad genera una notificación de que la prueba fue realizada. En los pasos 604, 608, y 610 pueden involucrar el despliegue o almacenamiento de una indicación en el sistema de la computadora designada de que la prueba fue satisfecha, y/o transmitiendo tal indicación a un sistema de cómputo remoto para el despliegue o registro allí.

[0053] En el paso 605, la facilidad realiza la prueba de aprobación especificada por el parche. En algunas personalizaciones, el paso 605 involucra la llamada de un grupo de rutinas normales utilizada spor la facilidad para probar. En el paso 606, si la prueba realizada en el paso 605 fue satisfecha, entonces la facilidad continúa en el paso 601, sino la facilidad continúa en el paso 607. En el paso 607, si la notificación de resultado de prueba es habilitado para el parche, entonces la facilidad continúa en el paso 608, sino la facilidad continúa en el paso 609. En el paso 608, la facilidad genera una notificación de que la prueba no fue satisfecha. En el paso 609, si el manejo del resultado de la prueba se habilita para el parche, entonces la facilidad continúa en el paso 610, sino, la facilidad continúa en paso 601. En paso 610, la facilidad realiza el prueba resultado manejando especificó por el parche. Después del paso 610, la facilidad continúa en el paso 601.

[0054] Será apreciado por estos experimentado en el arte que la facilidad arriba descrita puede adaptarse correctamente o puede extenderse de varias maneras. Por ejemplo, la facilidad puede usarse para aplicar una

columna del parche 418 que contiene un apuntador al propio parche, especificando una prueba y un manejo de resultado de la prueba para ser realizado cada vez que falla la prueba. En algunas personalizaciones, en lugar de contener un indicador como el parche mostrado, la columna del parche 418 contiene cada parche directamente. Una tabla de parche particular puede contener o puede apuntar a parches de varios tipos, como todos los parches de manejo de código, todos los parches de manejo de datos, o una combinación de parches de manejo de código y de manejo de datos.

[0050] En el paso 305, una vez la facilidad ha agregado el parche recibido a la tabla de parches y ha inicializado sus configuraciones, el parche está disponible para ser aplicado automáticamente por la facilidad en el módulo ejecutable. En el paso 305, la facilidad puede utilizar una variedad de acercamientos para aplicar el parche, incluyendo aquéllos descritos en la aplicación incorporada por la referencia, así como la función de tiempo real llamada interceptación, y/o la reescritura de código de (1) módulos ejecutables ya listos cargados, (2) uno o más imágenes del módulo ejecutable del disco, o (3) instancias de módulos ejecutables cargados por el cargador del sistema operativo. Después del paso 305, la facilidad continúa en el paso 301 para recibir el próximo parche.

[0051] La figura 5 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para poner al día las instrucciones de la configuración para un parche en particular. En el paso 501, la facilidad recibe las instrucciones de la configuración para un parche en particular, tal como de un administrador. En algunas personalizaciones, tales instrucciones de configuración pueden ser generadas por administradores que usan las políticas de grupo. En el paso 502, la facilidad actualiza que el parche es la configuración de la Tabla de parches, de acuerdo con las instrucciones recibidas. Después del paso 502, la facilidad continúa en el paso 501 para recibir las próximas instrucciones de configuración.

variedad de tipos diferentes de parches en una variedad de maneras a una variedad de situaciones en los módulos ejecutables de una variedad de tipos para una variedad de propósitos. También, mientras son descritos los parches aquí dentro de cómo contener pruebas de aprobación de valor que indican un problema cuando ellos fallan, la facilidad también puede ser implementada usando la pruebas de invalidez de valor que indican un problema que puede llevarse a cabo cuando ellos tienen éxito. En algunas personalizaciones, cada prueba se acompaña por una indicación de si el éxito o el fracaso indica un problema. Mientras la descripción anterior hace referencia a las personalizaciones preferidas, el alcance de la invención se define solamente por las demandas que siguen y los elementos citados aquí dentro.

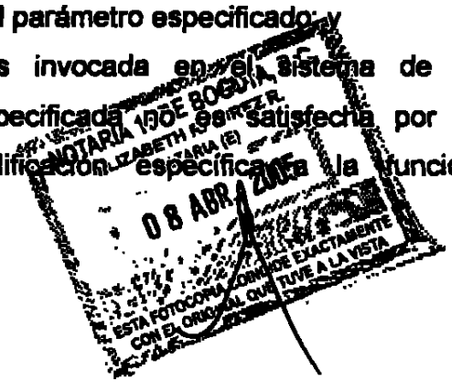
LAS DEMANDAS

Nosotros Demandamos:

[c1] 1. Un método en un sistema de cómputo para aumentar el software, comprendiendo:

Recibir en un sistema de cómputo designado una especificación de aumento que especifica (a) una función a ser aumentada, la función especificada estando entre el software ejecutado en el sistema de cómputo, (b) un parámetro de la función a ser aprobado, (c) una prueba para aplicar al parámetro especificado, y (d) una modificación para realizar al comportamiento de función si la prueba especificada no es satisfecha por el parámetro especificado; y

Cuando la función especificada es invocada en el sistema de la computadora designada, si la prueba especificada no es satisfecha por el parámetro especificado, realizar la modificación especificada a la función especificada.



[c2] 2. El método de la demanda 1 en donde la modificación especificada por la especificación de aumento está previniendo la ejecución de la función especificada.

[c3] 3. El método de la demanda 1 de la modificación especificada por la especificación de aumento está ejecutando la función especificada después de la alteración del parámetro especificado.

[c4] 4. El método de la demanda 1, en donde una pluralidad de especificaciones de aumento se recibe en el sistema de la computadora designada.

[c5] 5. El método de la demanda 4, más allá comprende el manteniendo de todas las especificaciones de aumento recibidas en una estructura de datos.

[c6] 6. El método de la demanda 1 en qué ninguna intervención de usuario subsiguiente se requiere para recibir la especificación de aumento para realizar la modificación especificada a la conducta de la función especificada.

[c7] 7. El método de la demanda 1 en donde la especificación de aumento especifica una prueba para aplicar al parámetro especificado identificando un parámetro que prueba la función para invocar de quien el código no es incluido en la especificación de aumento.

[c8] 8. El método de la demanda 1 en donde la especificación de aumento especifica una prueba a ser aplicada al parámetro especificado identificando un parámetro que prueba la función para invocar que fue instalado antes de la especificación de aumento recibida.

91

[c9] 9. El método de la demanda 1 en donde la especificación de aumento contiene el código.

[c10] 10. El método de la demanda 9 en donde el código contenido por la especificación de aumento incluye código que invoca una función de prueba de parámetro cuyo código no es incluido en la especificación de aumento.

[c11] 11. El método de la demanda 1 en donde la especificación de aumento contiene texto que identifica una función de prueba de parámetro cuyo código no está incluido en la especificación de aumento.

[c12] 12. El método de la demanda 11 en donde una función de prueba de parámetro identificada por el texto contenido por la especificación de aumento está contenida por la especificación de aumento.

[c13] 13. El método de la demanda 11 en donde una función de prueba de parámetro identificada por el texto no contenido por la especificación de aumento no está contenida por la especificación de aumento.

[c14] 14. El método de la demanda 1 más allá comprende proporcionar una interfaz para configurar un estado para controlar la operación de la especificación de aumento,
y en donde la modificación especificada para el comportamiento de la función especificada es realizada solo cuando el estado es un estado habilitado.

[c15] 15. El método de la demanda 14 en donde la interfaz proporcionada permite que la operación de la especificación de aumento sea configurada localmente.

92.

[c16] 16. El método de la demanda 14 en donde en donde la interfaz proporcionada permite la operación de la especificación de aumento sea configurada remotamente.

[c17] 17. El método de la demanda 14 en donde donde la interfaz proporcionada permite la operación de la especificación de aumento sea configurada de acuerdo con una política.

[c18] 18. El método de la demanda 14 en donde la prueba especificada es realizada antes de que cualquier código sustantivo de la función especificada se ejecute.

[c19] 19. El método de la demanda 1 más allá comprende proporcionar una interfaz para la configurando de advertencias asociadas con la especificación de aumento, y donde, si la interfaz proporcionada es usada para habilitar las advertencias, generando una advertencia cada vez que la prueba especificada por la especificación de aumento falla.

[20] 20. El método de la demanda 1, más allá comprende proporcionar una interfaz para configurar las notificaciones asociadas con la especificación de aumento, y en donde, si la interfaz proporcionada se usa para habilitar las notificaciones, generando una notificación cada vez que la prueba especificada por la especificación de aumento es realizada.

[c21] 21. El método de la demanda 20, más allá comprende consolidar las notificaciones generadas en una sola notificación consolidada.

[c22] 22. El método de la demanda 1 en donde la especificación de aumento recibida es firmada,
el método más allá comprende verificar la firma de la especificación de aumento,

93

y en donde la modificación especificada para el comportamiento de la función especificada sólo es realizado si la verificación de la firma de la especificación de aumento tuvo éxito.

[c23] 23. El método de la demanda 22 en donde la modificación especificada para el comportamiento de la función especificada es realizado solo si el autenticador de la firma de la especificación de aumento coincide con un autenticador del software que contiene la función especificada.

[c24] 24. El método de la demanda 1, más allá comprende realizar la prueba especificada en respuesta a descubrir que la función especificada ha sido invocada, sin alterar el código de la función especificada.

[c25] 25. El método de la demanda 1, más allá comprende alterar el código de la función especificada para realizar la prueba especificada cuando la función especificada es invocada.

[c26] 26. El método de la demanda 25 en donde el código de la función especificada es alterada insertando en el código de la función especificada un salto a una rutina separada que realiza la prueba especificada.

[c27] 27. El método de la demanda 1 en donde la función especificada es proporcionada en un módulo ejecutable distinguido que es posible cargar usando a un cargador, el método más allá comprende registrar con el cargador para tener una rutina que realiza el llamado de la prueba especificada antes de que la función especificada sea ejecutada en respuesta a cada invocación de la función especificada.

[c28] 28. Un sistema de cómputo que habilita el aumento de software presente en el sistema de cómputo, comprendiendo:

un dispositivo de almacenamiento que contiene el software;

94

un receptor del parche que recibe en el sistema de cómputo un parche especificando (a) un punto al cual el software será aumentado, (b) un valor asociado con la función a ser probada en el punto especificado, (c) una prueba para aplicar al valor especificado, y (d) una modificación para realizar al comportamiento del software si la prueba especificada no es satisfecha por el valor especificado; y

un agente parches que inyecta el código en el software al punto especificado tal que, cuando la ejecución del software alcanza el punto especificado en el sistema de la computadora designada, si la prueba especificada no es satisfecha por el valor especificado, la modificación especificada al comportamiento del software es realizada.

[c29] 29. Un medio legible por computadora cuyos contenidos causan a un sistema de cómputo designado para realizar un método para adicionar la aprobación de valor del software disponible en el sistema de cómputo designado, comprendiendo:

recibir en el sistema de cómputo una especificación de aumento que especifica (a) una función para la cual la aprobación de valor será agregada, la función especificada que está entre el software disponible y el sistema de cómputo, (b) los datos a ser probados que existen en el sistema de cómputo designado durante la ejecución de la función, (c) una prueba para aplicar a los datos especificados, y (d) una modificación para realizar al comportamiento de la función si la prueba especificada no es satisfecha por los datos especificados; y

cuando la función especificada se invoca en el sistema de la computadora designada, si la prueba especificada no es satisfecha por los datos especificados, realizando la modificación especificada al comportamiento de la función especificada.

[c30] 30. El medio legible por la computadora de demanda 22 en donde la prueba especificada es una prueba que siempre es satisfecha, independientemente del valor de los datos especificados.

95

[c31] 31. Una señal de datos generada llevando un parche de la estructura de datos, comprendiendo:

información que identifica una función a la cual la aprobación de valor será agregada;

información que identifica datos a ser probados que existen durante la ejecución de la función;

información que identifica una prueba para aplicar a los datos especificados; y para

información que identifica una modificación para realizar al comportamiento de la función si la prueba especificada no es satisfecha por los datos especificados,

Así, si la señal de datos es recibida en un sistema de la computadora designada en el cual la función es ejecutada, los contenidos de la estructura de datos puede ser usada para realizar la modificación especificada al comportamiento de la función especificada si la prueba especificada no es satisfecha por los datos especificados cuando la función especificada se invoca en el sistema de la computadora designada.

[c32] 32. La señal de datos generada de la demanda 31 en donde la señal de datos generada se transmite entre los sistemas de cómputo.

[c33] 33. La señal de datos generada de la demanda 31 en donde La señal de datos generada se transmite dentro de un sistema de cómputo.

[c34] 34. Un método para agregar la aprobación de valor al software disponible en el sistema de cómputo designado, comprendiendo:

recibir en el sistema de cómputo una especificación de aumento que especifica (a) software para el cual la aprobación de valor será agregada, (b) un punto en el software especificado en el cual la aprobación de valor será agregada, (c) los datos a ser probados que existen en el sistema de la computadora designada durante la ejecución del software especificado en

96

el punto especificado, (d) una prueba para aplicar a los datos especificados, y (e) una modificación para realizar al comportamiento del software si la prueba especificada no está satisfecha por los datos especificados;

permitir a un usuario que configure un modo de operación para la especificación de aumento recibida; y

cuando el software especificado se ejecute en el punto especificado en el sistema de la computadora designada, aplicando la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida.

[c35] 35. El método de la demanda 34 en donde el usuario configura un modo operacional activo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende, si la prueba especificada no está satisfecha por los datos especificados, realizando la modificación especificada al comportamiento del software especificado.

[c36] 36. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico activo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende, si la prueba especificada no está satisfecha por los datos especificados, ambos (1) realizar la modificación especificada al comportamiento del software especificado, y (2) generar una notificación de que la prueba especificada no es satisfecha por los datos especificados.

[c37] 37. El método de la demanda 34 en donde el usuario configura un modo operacional activo difuso para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende

94

si la prueba especificada no es satisfecha por los datos especificados, realizando la modificación especificada al comportamiento del software especificado; y

generar una notificación de que la prueba especificada fue realizada.

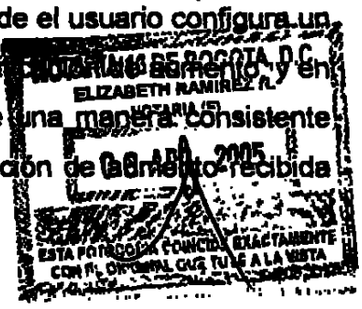
[c38] 38. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico difuso activo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

si la prueba especificada no es satisfecha por los datos especificados, ambos (1) realizar la modificación especificada a la conducta del software especificado, y (2) generar una notificación de que la prueba especificada no es satisfecha por los datos especificados; y

generar una notificación de que la prueba especificada fue realizada.

[c39] 39. El método de la demanda 34 en donde el usuario configura un modo operacional inactivo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados.

[c40] 40. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico inactivo l para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:



omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados; y

si la prueba especificada no está satisfecha por los datos especificados, generando una notificación de que la prueba especificada no es satisfecha por los datos especificados.

[c41] 41. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico inactivo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

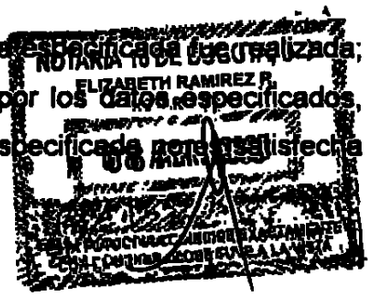
omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados; y

generar una notificación de que la prueba especificada fue realizada.

[c42] 42. El método de la demanda 34 en donde el usuario configura modo operacional de diagnóstico inactivo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados;

generar una notificación de que la prueba especificada fue realizada; y si la prueba especificada no es satisfecha por los datos especificados, generando una notificación de que la prueba especificada no es satisfecha por los datos especificados.



ABSTRACTO

Una facilidad para el software de aumento en un sistema de cómputo designado es descrito. La facilidad recibe y la especificación de aumento en el sistema de la computadora designada. La especificación de aumentos especifica: (a) una función a ser aumentada, (b) un parámetro de la función a ser probado, (c) una prueba para aplicar al parámetro especificado, y (d) y la modificación para realizar al comportamiento de la función si la prueba especificada no es satisfecha por el parámetro especificado. Cuando la función especificada es invocada en el sistema de la computadora designada, si lo probado especificado no es satisfecho por el parámetro especificado, la facilidad realiza la modificación especificada al comportamiento de la función especificada.

TRADUCCIÓN PARCIAL DEL INGLÉS AL ESPAÑ L DE CINCO PÁGINAS DE UN DOCUMENTO ENTREGADO EN UN F LDER DE LA FICINA DE PATENTES Y MARCAS REGISTRADAS DEL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS, CON COBERTURA DE PLÁSTICO TRANSPARENTE, REALIZADA POR NORA JIMÉNEZ DE ADAM, TRADUCTORA E INTÉRPRETE OFICIAL. TEL: 571-6917116; 30 DE MARZO DE 2005

PRIMERA PÁGINA: Encabezado de envío de fax 4 de junio de 2004 23:02 DE: A: 92063599000, P.02-19

CORREO EXPRESO N° EV336672580US

SOLICITUD DE PATENTE

DECLARACIÓN Y PODER

Expediente del Abogado No.41826.8010US1 Expediente de MS No. 307667.26

Como el inventor nombrado a continuación, por medio del presente declaro que:

Mi residencia/dirección postal y ciudadanía son los indicados a continuación debajo de mi nombre;

Creo ser el inventor original, primero y único (en caso de haber sólo un nombre indicado abajo), o el inventor original, primero y en conjunto (en caso de haber varios nombres indicados abajo) del objeto reivindicado y por el cual se solicita una patente sobre el invento llamado **PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE**

cuya especificación se presenta anexa al presente, salvo que la siguiente casilla se encuentra marcada:

( ) fue presentado el ____ como Número de Serie de Solicitud de Estados Unidos o Número de Solicitud Internacional PCT _____ modificado el _____ (si aplica). Declaro que he revisado y entendido los contenidos de la especificación previamente mencionada, incluyendo las reivindicaciones, en lo modificado por la (s) modificación (es) referida (s) arriba. Reconozco el deber de revelar toda información esencial a la patentabilidad según lo definido en 37 C.F.R. CFR 1.56.

ELIZABETH RAMIREZ R.
NOTARIA (E)
08 APR. 2005
CON EL ORIGINAL QUE TUVO A LA VISTA

Solicitud (es) Extranjeras y/ Reivindicación de Prioridad en el Extranjero

Pro el presente reivindico los beneficios de prioridad en el extranjero según lo siguiente

Título 35 de la sección 119 del Código de los Estados Unidos, sobre cualquier solicitud de patente o certificado de inventor indicado a continuación y he identificado a continuación cualquier solicitud de patente o inventor (es) certificados con una fecha de presentación previa a la solicitud sobre el cual se reivindica la prioridad:

PAÍS	NO. SOLICITUD	FECHA DE PRESENTACIÓN	PRIORIDAD REIVINDICADA SEGÚN 35 U.S.C. 119
			SI: ___ NO:
			SI: ___ NO:

PODER:

Como inventor, nombro por medio del presente el (los) siguiente (s) abogado (s) y/o agente (s) asociados con

Cliente N° 25096

P.O. Box 1247

Seattle, Washington 98111-1247

para hacer seguimiento a esta solicitud y llevar a cabo todas las transacciones con la Oficina de Patentes y Marcas Registradas relacionadas con el mismo. STEPHEN E. ARNETT, Registro N° 47,392; RODGER K. CARREYN, Registro N° 50.774; MAY Y. CHAN, Registro N° 51,053; BRIAN R. COLEMAN, Registro N° 59,145; CHRISTOPHER DALEY-WATSON, Registro N° 34,807; PETER J, DEHLINGER, Registro N° 28,006; DAVID DEVLIN, Registro N° 65,876; DAVID BOGART DORT, Registro N°50,219; DAVID T. DUTCHER, Registro N° 51,638; LEEANN CORTHEY, Registro N° 37,337; JOSEPH HAMILTON, Registro N° 51,770; PAUL L. HICKMAN, Registro N° 28,516; EDWARD S. HOTCHKISS, Registro N° 33,904; STEVEN KELLEY, Registro N° 45,449;DO TE KIM, Registro N° 46,231; JONATHAN P. KUDLA, Registro N° 47,724; STEVEN D. LAWRENZ, Registro N° 37,376; JACQUELINE P. MAHONEY, Registro N° 48,390; SHAILESH MERA, Registro N° 44,934; JUDY M. MOHR, Registro N° 38,563; CHUN M. NC, Registro N° 38,264; KENNETH J. OBERNEIER, Registro N° 31,648; PAUL T. PARKER, Registro N° 38,264; MAURICE J. PIRIO, Registro N° 33,273; AARON J. POLEDNA, Registro N° 54,675; JAJIV P. SARATHY, Registro N° 55,592; MICHELLE SARRUB, Registro N° 55,828; TIM R. SEELEY, Registro N° 53,579; LAUREN SEIGER, Registro N° 51,986; CARINA M. TAN, Registro N° 45,769; LARRY W. THROWER, Registro N°

NOTARIA 16 DE LOJA
ELIZABETH RAMIREZ R.
NOTARIA
08 ABR. 2005
ESTA FOTOCOPIA ES IGUAL AL QUE TUVE A LA VISTA
CON EL ORIGINAL

47,994; GLENN P. VON TERSH, Registro N° 41,364; JOHN M. WECHKIN, Registro N° 42,216; MICHAEL J. WISE, Registro N° 34,047; ROBERT G. WOOLSTON, Registro N° 37,263; JAMES J. ZHU, Registro N° 52,996; todos afiliados a Perkins Cole LLP, con PATRICA E. BORNES, Registro N° 37,038; DAVID BARTLEY EPPENAUER, Reg. N° 35,499; STACY QUAN, Registro N° 33,760; JEFFREY L. RANCK, Registro N° 38,590; MARTIN L. SHIVELY, Registro N° 33,553 y JOHN WERESH, Registro N° 32,322; de Microsoft Corporation, One Microsoft Way, Redmond, Washington 98062 como principales abogados con poder absoluto de sustitución, asociación y revocación para procesar dicha solicitud, hacer transacciones de todos los negocios en conexión con esta Oficina de Patentes y marcas Registradas y para recibir las patentes del invento aquí citado.

Enviar correspondencia a:	Dirigir llamadas a:
Contacto:	Steven D. Lawrenz
Nombre de la Firma:	Perkins Cole LLP
	Teléfono de Contacto: (206) 359-8000
Dirección de la Firma:	P.O. Box 1247
Ciudad, Estado y Código Postal:	Seattle, WA 98111-1247

SIGUIENTE PÁGINA:

DECLARACIÓN Y PODER

Expediente del Abogado No. 41826.8010US1 Expediente de MS No. 307667.26

Declaro que todas las afirmaciones aquí establecidas son en mi conocimiento verdaderas y que todas las afirmaciones están basadas en informaciones y creencias que se cree son verdaderas; adicionalmente, todas las afirmaciones fueron elaboradas con el conocimiento de que declaraciones falsas intencionales y relacionadas son castigados con multa o prisión o ambos, según Sección 1001 del Título 18 del Código de los Estados Unidos, y que dichas declaraciones falsas intencionales pueden poner en peligro la validez de

la solicitud o cualquier patente otorgado por ésta.

104

Nombre Completo del Inventor: Anthony Blumfield Ciudadanía: Inglés
Residencia: Redmond, Washington
Dirección de la Oficina: 18229 NE 28th. Street, Redmond Washington 98052
Fecha: 8/8/03
(Firma) Anthony Blumfield

Nombre Completo del Inventor: Gilad Golan Ciudadanía: Israel
Residencia: Redmond, Washington
Dirección Postal: 16908 NE 43rd. Ct. Redmond, Washington 98052
Fecha: (en blanco)
(Firma en blanco) Gilad Golan

Nombre Completo del Inventor: Jason Garms Ciudadanía: E.U.
Residencia: Woodinville, Washington
Dirección Postal de la Oficina: 12014 198th. Ct. NE Woodinville, Washington 98077 Fecha: (en blanco)
(Firma en blanco) Jason Garms

DECLARACIÓN Y PODER

Expediente del Abogado No. 41826.8010US1 Expediente de MS No. 307667.26

Declaro que todas las afirmaciones aquí establecidas son en mi conocimiento verdaderas y que todas las afirmaciones están basadas en información y creencias que se cree son verdaderas; adicionalmente, todas las afirmaciones fueron elaboradas con el conocimiento de que declaraciones falsas intencionales y relacionadas son castigados con multa o prisión o ambos, según Sección 1001 del Título 18 del Código de los Estados Unidos y que

NOTARIA 16 DE BOGOTA D.C.
ELIZABETH RAMIREZ
NOTARIA (E)
0 8 ABR 2005
ESTE FOTOCOPIA CORRESPONDE EXACTAMENTE

184

dichas declaraciones falsas intencionales pueden poner en peligro la validez de la solicitud o cualquier patente otorgado por ésta.

Nombre Completo del Inventor: Anthony Blumfield Ciudadanía: Inglés
Residencia: Redmond, Washington
Dirección de la Oficina: 18229 NE 26th. Street, Redmond Washington 98052
Fecha: 8/8/03
(Firma en blanco) Anthony Blumfield

Nombre Completo del Inventor: Gilad Golan Ciudadanía: Israel
Residencia: Redmond, Washington
Dirección Postal: 16908 NE 43rd. Ct. Redmond, Washington 98052
Fecha: 6/4/04
(Firma) Gilad Golan

Nombre Completo del Inventor: Jason Garms Ciudadanía: E.U.
Residencia: Woodinville, Washington
Dirección Postal de la Oficina: 12014 198th. Ct. NE Woodinville, Washington 98077 Fecha: (en blanco)
(Firma en blanco) Jason Garms

SIGUIENTE PÁGINA:

DECLARACIÓN Y PODER

Expediente del Abogado No. 41826.8010US1 Expediente de MS No. 307667.26

Declaro que todas las afirmaciones aquí establecidas son en mi conocimiento verdaderas y que todas las afirmaciones están basadas en información y creencias que se cree son verdaderas; adicionalmente, todas las afirmaciones fueron elaboradas con el conocimiento de que declaraciones falsas intencionales y relacionadas son castigados con multa o prisión o ambos, según Sección 1001 del Título 18 del Código de los Estados Unidos, y que

dichas declaraciones falsas intencionales pueden poner en peligro la validez de la solicitud o cualquier patente otorgado por ésta.

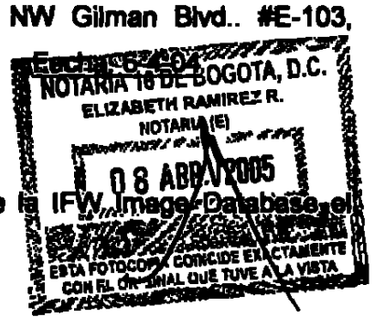
Nombre Completo del Inventor: Anthony Blumfield Ciudadanía: Inglés
Residencia; Redmond, Washington
Dirección de la Oficina: 18229 NE 28th. Street, Redmond Washington 98052
Fecha: 8/8/03
(Firma en blanco) Anthony Blumfield

Nombre Completo del Inventor: Gilad Golan Ciudadanía: Israel
Residencia: Redmond, Washington
Dirección Postal: 16908 NE 43rd. Ct. Redmond, Washington 98052
Fecha: 6/4/04
(Firma en blanco) Gilad Golan

Nombre Completo del Inventor: Jason Garms Ciudadanía: E.U.
Residencia: Woodinville, Washington
Dirección Postal de la Oficina: 12014 198th. Ct. NE Woodinville, Washington 98077 Fecha: junio 24 de 2004
(Firma) Jason Garms

SIGUIENTE PÁGINA:

Nombre Completo del Inventor: Saud Alshibani Ciudadanía: E.U.
Residencia: Issaquah, Washington
Dirección Postal de la Oficina: pmb 152, 700 NW Gilman Blvd.. #E-103, Issaquah, Washington 98027-5335 Fecha: 6-4-04
(Firma) Saud Alshibani



104

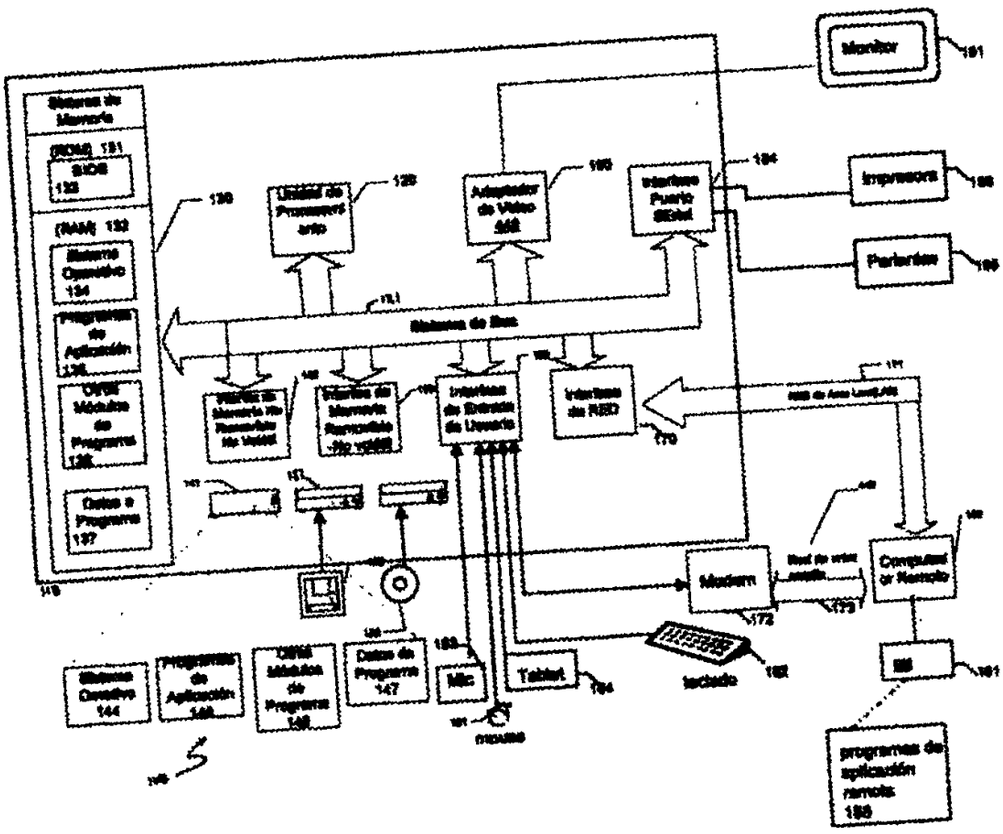


Fig. 1

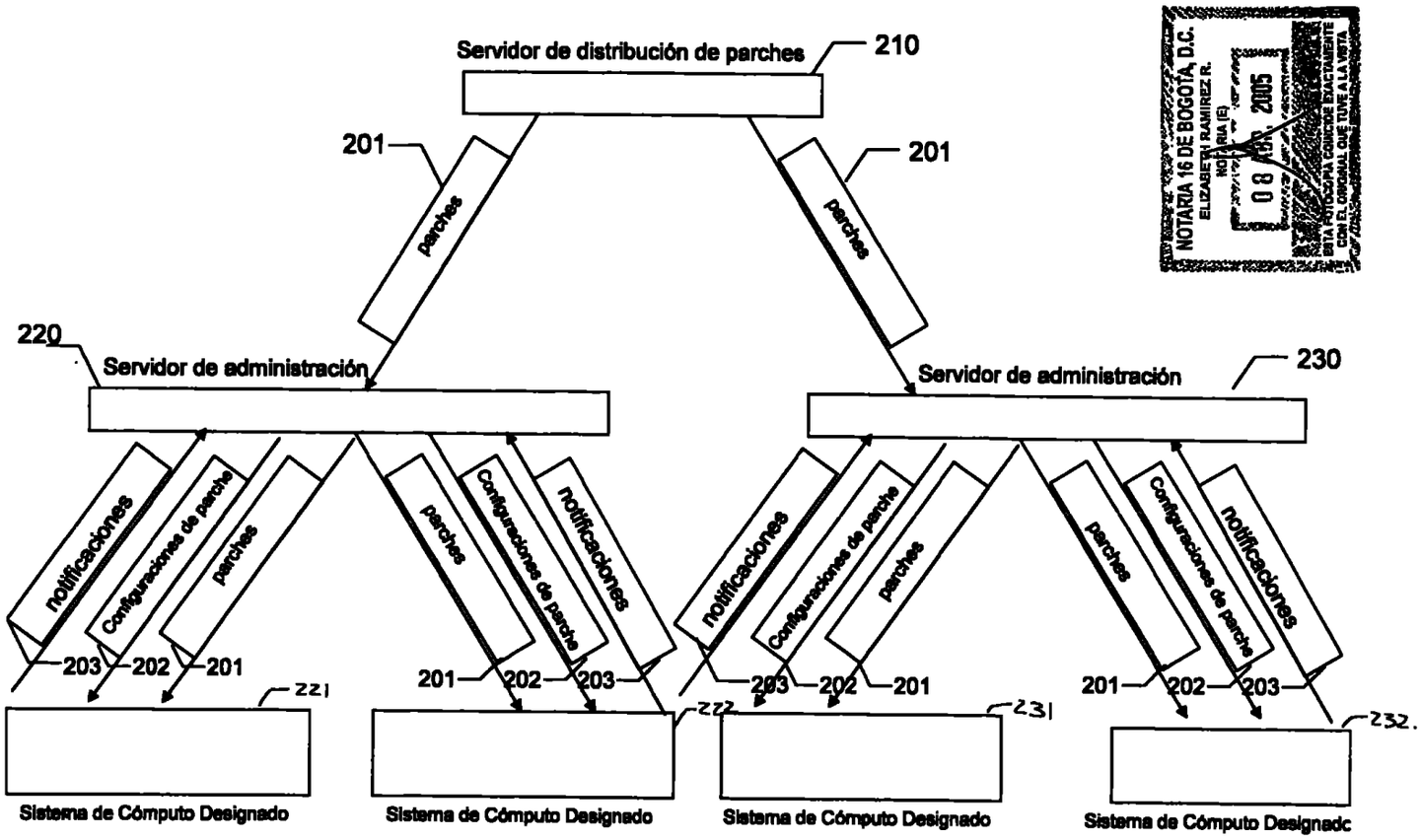


Fig. 2

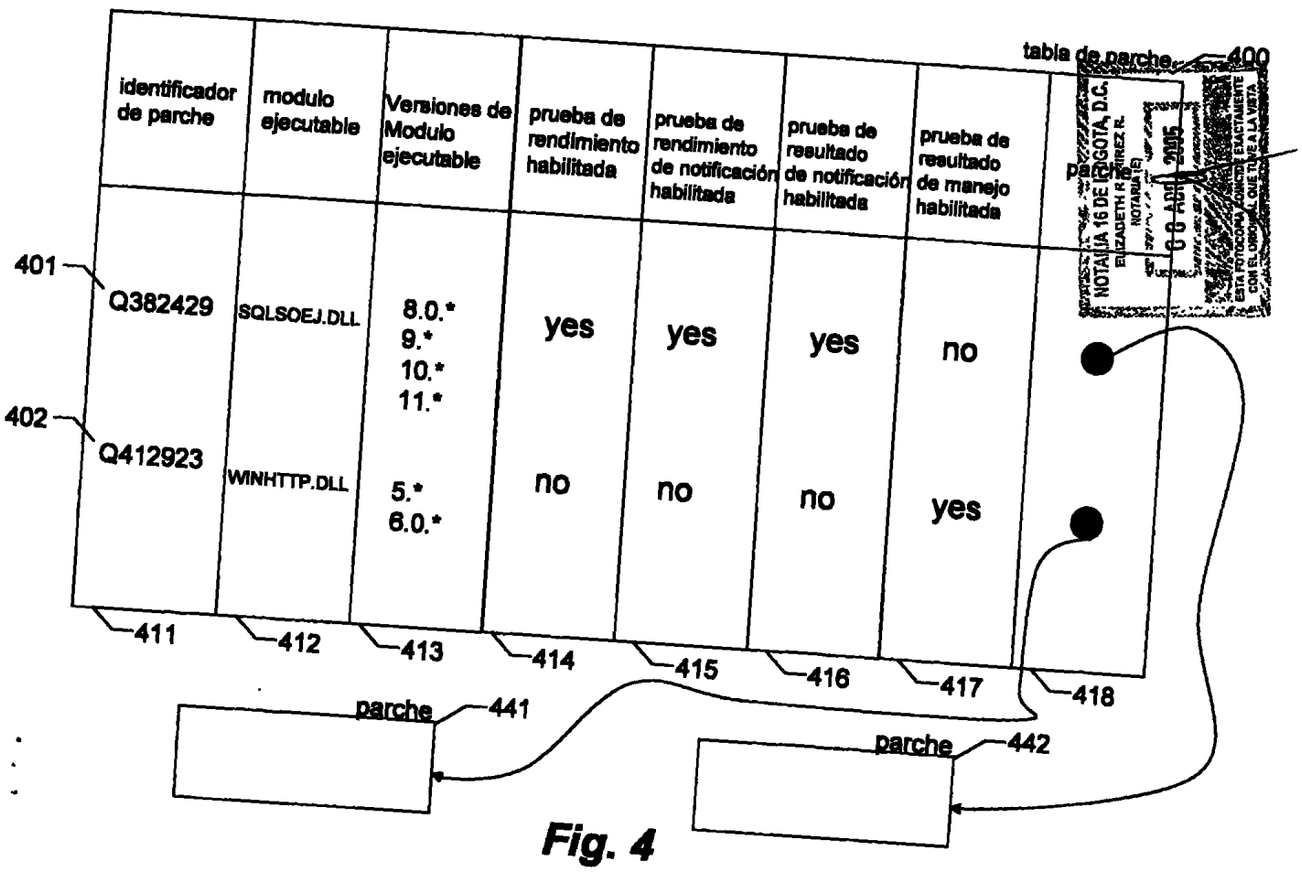


tabla de parche
400
identificador de parche
modulo ejecutable
Versiones de Modulo ejecutable
prueba de rendimiento habilitada
prueba de rendimiento de notificación habilitada
prueba de resultado de notificación habilitada
prueba de resultado de manejo habilitada
parche
401
Q382429
SQLSOEJ.DLL
8.0.*
9.*
10.*
11.*
yes
yes
yes
no
402
Q412923
WINHTTP.DLL
5.*
6.0.*
no
no
no
yes
411
412
413
414
415
416
417
418
parche
441
parche
442

Fig. 4

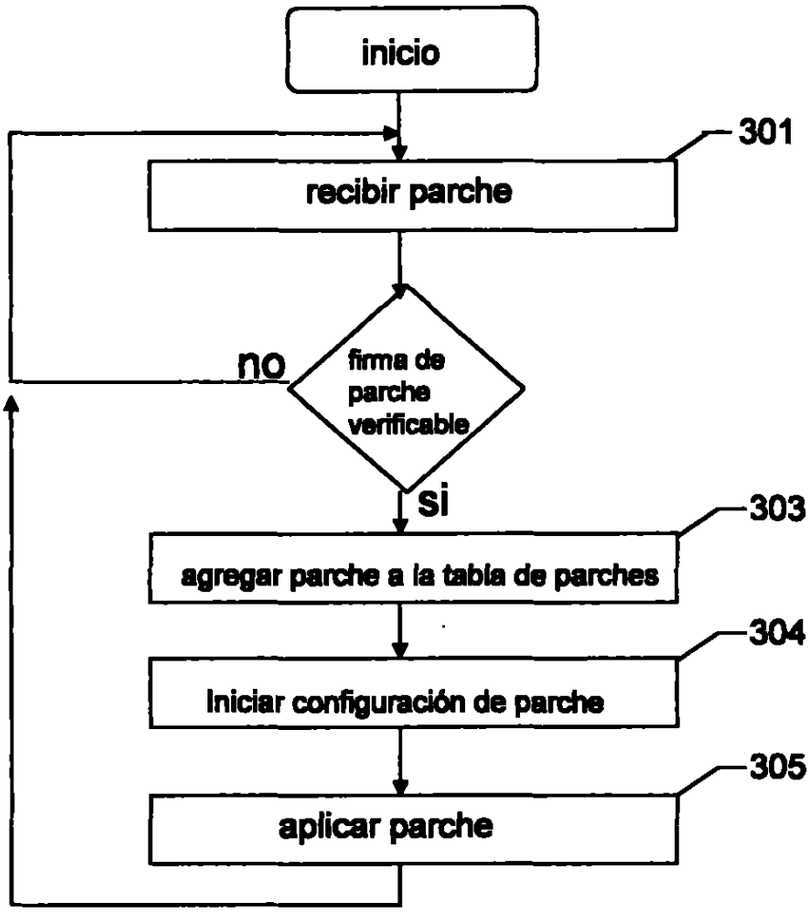


Fig. 3

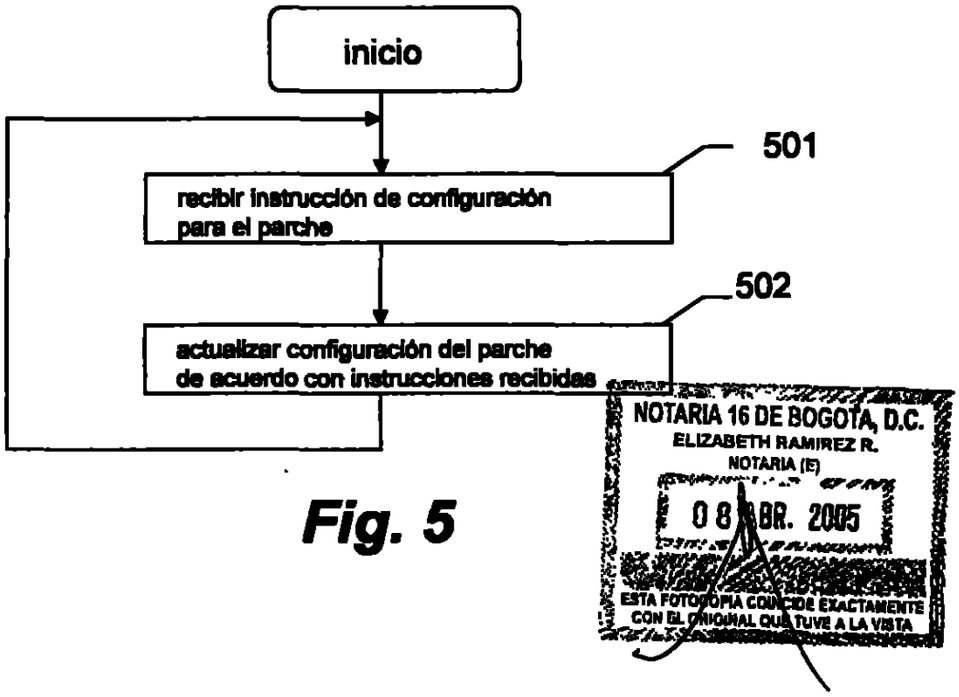


Fig. 5

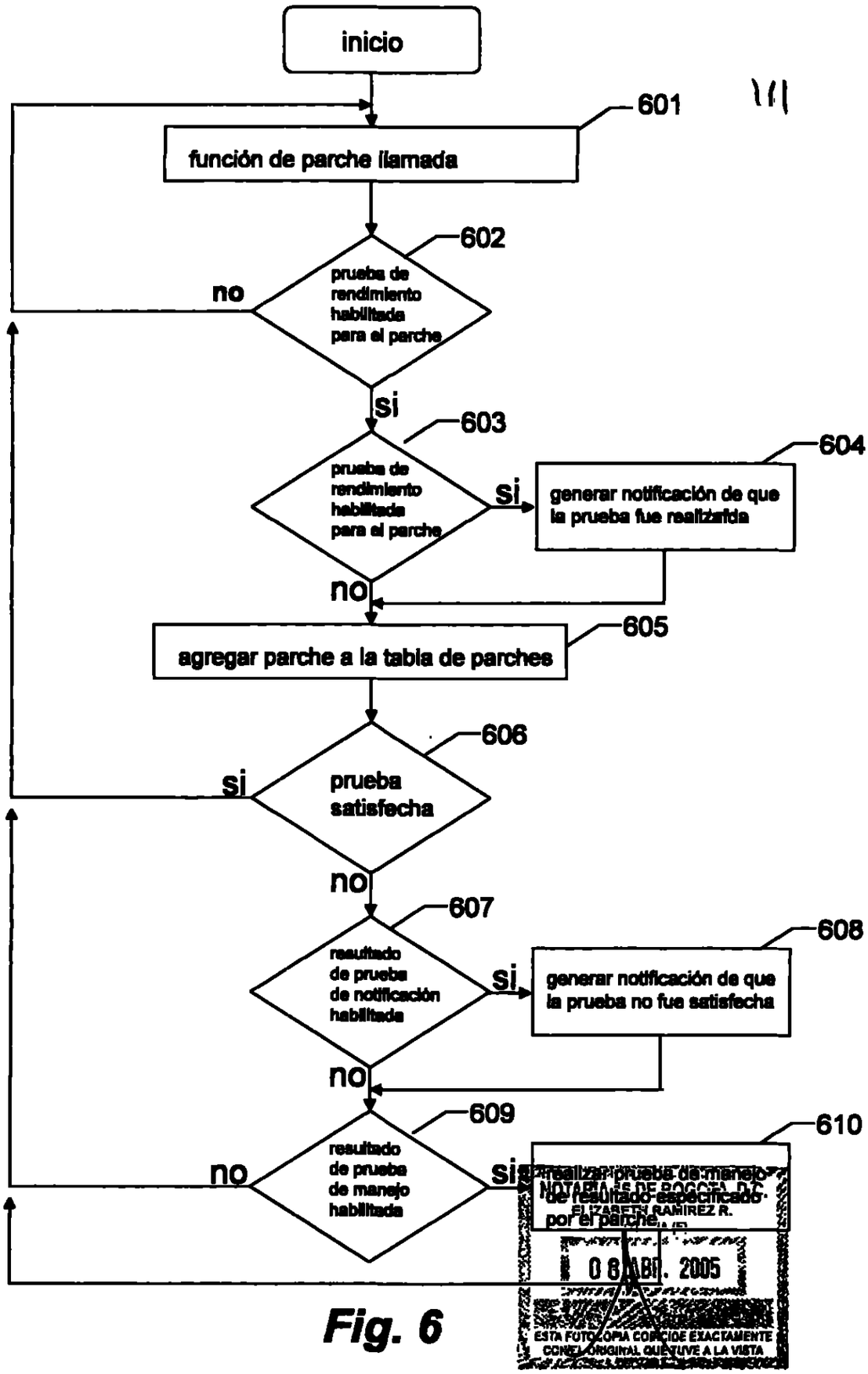

inicio
601
función de parche llamada
602
prueba de rendimiento habilitada para el parche
no
si
603
prueba de rendimiento habilitada para el parche
si
604
generar notificación de que la prueba fue realizada
no
605
agregar parche a la tabla de parches
606
prueba satisfecha
si
no
607
resultado de prueba de notificación habilitada
si
608
generar notificación de que la prueba no fue satisfecha
no
609
resultado de prueba de manejo habilitada
no
si
610
realizar prueba de manejo de resultado especificado por el parche
NOTARIA 35 DE BOGOTA D.C.
ELIZABETH RAMIREZ R.
06 ABR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

Fig. 6

1
112

PA 1251798

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

November 24, 2004

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: ***10/880,709***
FILING DATE: ***June 30, 2004***

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

E. BORNETT
Certifying Officer

NOTARIA 16 DE BOGOTA D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 MAR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE

TPC

113

PTO/SB/05 (04-04)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(ONLY FOR NEW NONPROVISIONAL APPLICATIONS UNDER 37 CFR 1.53(b))

Attorney Docket No.	41626.8010US1
First Inventor	Anthony Blumfield
Title	EFFICIENT PATCHING
Express Mail Label No.	EV338672580US

15866 U.S.PTO 083004

12553 U.S. PTO 10/890709 083004

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

1. [X] Fee Transmittal Form (e.g., PTO/SB/17) (Submit an original and a duplicate for fee processing)
2. [ ] Applicant claims small entity status. See 37 CFR 1.27.
3. [X] Specification [Total Pages 31]
   (preferred arrangement set forth below)
   - Descriptive title of the invention
   - Cross References to Related Applications
   - Statement Regarding Fed sponsored R & D
   - References to sequence listing, a table, or a computer program listing appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings (if filed)
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
4. [X] Drawing(s) (35 U.S.C. 113) [Total Sheets 5]
5. Oath or Declaration [Total Sheets 5]
   a. [X] Newly executed (original or copy)
   b. [ ] Copy from a prior application (37 CFR 1.63(d)) (for continuation/divisional with Box 16 completed)
      i. [ ] DELETION OF INVENTOR(S) SIGNED STATEMENT ATTACHED DELETING INVENTOR(S) NAMED IN THE PRIOR APPLICATION, SEE 37 CFR 1.63(d)(2) AND 1.33(b).
6. [X] Application Data Sheet. See 37 CFR 1.76
7. [ ] CD-ROM or CD-R in duplicate, large table or Computer Program (Appendix)
8. Nucleotide and/or Amino Acid Sequence Submission (If applicable, all necessary)
   a. [ ] Computer Readable Form (CRF)
   b. Specification Sequence Listing on:
      i. [ ] CD-ROM or CD-R (2 copies); or ii. [ ] Paper
   c. [ ] Statements verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

9. [X] Assignment Papers (cover sheet & document(s))
10. [ ] 37 CFR 3.73(b) Statement (when there is an assignee) [ ] Power of Attorney
11. [ ] English Translation Document (if applicable)
12. [X] Information Disclosure Statement (IDS)/PTO-1449 [X] Copies of IDS Citations
13. [ ] Preliminary Amendment
14. [X] Return Receipt Postcard (MPEP 503) (Should be specifically itemized)
15. [ ] Certified Copy of Priority Document(s) (if foreign priority is claimed)
16. [ ] Nonpublication Request under 35 U.S.C. 122 (b)(2)(B)(i). Applicant must attach form PTO/SB/35 or its equivalent.
17. [X] Other: Check

18. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in the first sentence of the specification following the title, or in an Application Data Sheet under 37 CFR 1.76:

[ ] Continuation [ ] Divisional [ ] Continuation-in-part (CIP) of prior application No.: _____

Prior application information: Examiner _____ Art Unit: _____

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

19. CORRESPONDENCE ADDRESS

[X] Customer Number: 25096 OR [ ] Correspondence address below

Name					
Address					
City		State		Zip Code	
Country		Telephone		Fax	

Name (Print/Type)	Stephen R. Lewinz	Registration No. (Attorney/Agent)	37,376
Signature		Date	6/30/04

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 APR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL ORIGINAL QUE TUVO A LA VISTA

TB

114

PTO/SB/17 (10-03)
Approved for use through 7/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL for FY 2004

Effective 10/01/2003. Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 1,378.00

Complete If Known	
Application Number	Not Yet Assigned
Filing Date	Concurrently Herewith
First Named Inventor	Anthony Blumfield
Examiner Name	Not Yet Assigned
Art Unit	N/A
Attorney Docket No.	41828.8010US1

METHOD OF PAYMENT (check all that apply)

☒ Check ☐ Credit Card ☐ Money Order ☐ Other ☐ None

☐ Deposit Account:

Deposit Account Number: 50-0665

Deposit Account Name: Perkins Coie LLP

The Director is authorized to: (check all that apply)

☐ Charge fee(s) indicated below ☒ Credit any overpayments

☒ Charge any additional fee(s) or any underpayment of fee(s)

☐ Charge fee(s) indicated below, except for the filing fee

to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	770	2001	385	Utility filing fee	770.00
1002	340	2002	170	Design filing fee	
1003	530	2003	265	Plant filing fee	
1004	770	2004	385	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	

SUBTOTAL (1) ($) 770.00

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

			Extra Claims		Fee from below		Fee Paid
Total Claims	42	-20** =	22	x	18.00	=	396.00
Independent Claims	5	-3** =	2	x	86.00	=	172.00
Multiple Dependent						=	

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	86	2201	43	Independent claims in excess of 3
1203	290	2203	145	Multiple dependent claims, if not paid
1204	86	2204	43	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 568.00

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge – late filing fee or oath	
1052	50	2052	25	Surcharge – late provisional filing fee or cover sheet.	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for ex parte reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	110	2251	55	Extension for reply within first month	
1252	420	2252	210	Extension for reply within second month	
1253	950	2253	475	Extension for reply within third month	
1254	1,480	2254	740	Extension for reply within fourth month	
1255	2,010	2255	1,005	Extension for reply within fifth month	
1401	330	2401	165	Notice of Appeal	
1402	330	2402	165	Filing a brief in support of an appeal	
1403	290	2403	145	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	110	2452	55	Petition to revive – unavoidable	
1453	1,330	2453	665	Petition to revive - unintentional	
1501	1,330	2501	665	Utility issue fee (or reissue)	
1502	480	2502	240	Design issue fee	
1503	640	2503	320	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	40.00
1809	770	2809	385	Filing a submission after final rejection (37 CFR 1.129(a))	
1810	770	2810	385	For each additional invention to be examined (37CFR 1.129(b))	
1801	770	2801	385	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	
Other fee (specify)					

*Reduced by Basic Filing Fee Paid SUBTOTAL (3) ($) 40.00

SUBMITTED BY

(Complete if applicable)

Name (Print/Type)	Steven D. Lawrence	Registration No. (Attorney/Agent) 37,376	Telephone (206) 359-8000
Signature			Date 6/30/04

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)

I hereby certify that this correspondence is being deposited with the U.S. Postal Service as Express Mail No. EV338572590US, in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below.

Dated: 6/30/04 Signature: (Victoria Carmody)

ESTA FOTOCOPIA CORRESPONDE CON EL ORIGINAL QUE TUVO A LA VISTA

11A

Express Mail No. EV336672580US

Application Data Sheet

Application Information

Application Type::	Regular
Subject Matter::	Utility
Suggested Group Art Unit::	N/A
CD-ROM or CD-R?::	None
Sequence submission?::	None
Computer Readable Form (CRF)?::	No
Title::	EFFICIENT PATCHING
Attorney Docket Number::	41826.8010US1
Request for Early Publication?::	No
Request for Non-Publication?::	No
Total Drawing Sheets::	5
Small Entity?::	No
Petition included?::	No
Secrecy Order in Parent Appl.?::	No

Applicant Information

Applicant Authority Type::	Inventor
Primary Citizenship Country::	British
Status::	Full Capacity
Given Name::	Anthony
Family Name::	Blumfield
Address:	18229 NE 28th Street, Redmond, WA 98052
Applicant Authority Type::	Inventor
Primary Citizenship Country::	Israel
Status::	Full Capacity
Given Name::	Gilad
Family Name::	Golan

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 ABR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

116

Address:	16908 NE 43rd Ct., Redmond, WA 98052
	Inventor
Applicant Authority Type::	
Primary Citizenship Country::	US
Status::	Full Capacity
Given Name::	Jason
Family Name::	Garms
Address:	12014 198th Ct. NE, Woodinville, WA 98077
	Inventor
Applicant Authority Type::	
Primary Citizenship Country::	US
Status::	Full Capacity
Given Name::	Saud
Family Name::	Alshibani
Address:	pmb 152, 700 NW Gilman Blvd, #E-103
	Issaquah, WA 98027-5335
Correspondence Information	
Correspondence Customer Number::	25096
Assignee Information	
Assignee Name:	Microsoft Corporation
Street of Mailing Address:	One Microsoft Way
City of Mailing Address:	Redmond
State or Province of Mailing Address:	WA
Postal or Zip Code of Mailing Address:	98052
Representative Information	
Representative Customer Number::	25096

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA
08 ABR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE
CON EL ORIGINAL QUE TUVE A LA VISTA

116

EXPRESS MAIL NO. EV336872580US

EFFICIENT PATCHING

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 60/570,124, entitled "EFFICIENT PATCHING," and filed on May 11, 2004 and is related to U.S. Patent Application No. 10/307,902, entitled "PATCHING OF IN-USE FUNCTIONS ON A RUNNING COMPUTER SYSTEM," and filed on December 2, 2002; U.S. Patent Application No. _____, (patent counsel's docket number 41826.8019US), entitled "EFFICIENT PATCHING", filed concurrently herewith; and U.S. Patent Application No. _____, (patent counsel's docket number 41826.8020US), entitled "EFFICIENT PATCHING", filed concurrently herewith, each of which is hereby incorporated by reference in its entirety.

[0002]

TECHNICAL FIELD

[0003] The present invention is directed to the field of updating the operation of installed computer programs.

BACKGROUND

[0004] Patching is the process of modifying already-installed programs, including application programs, utility programs, operating systems and operating system components, device drivers, etc. Patching can be useful to modify programs for a variety of purposes, including correcting a programming error, reducing or eliminating a security risk, or improving the logic used by the modified program. Patching is typically initiated by the company or other organization that originally supplied the program to be patched.

[0005] Installed programs are predominantly composed of executable code modules. As one example, many programs designed to execute on the WINDOWS XP operating system from Microsoft Corp. of Redmond, Washington are predominantly composed of executable code modules called "DLLs." One popular conventional approach to patching is to identify, among the executable code modules making up the installed program to be patched, the executable code module containing the program code that one wishes to modify with a patch; creating a new version of the identified executable code module in which the desired modification is made; and distributing the new version of the identified executable code module, together with an installer program, to users who may wish to apply the patch. Each user then determines whether s/he wishes to apply the patch, and if so, executes the installer program, which replaces the original version of the identified executable code module with the new version of the identified executable code module.

[0006] Conventional approaches to patching have a number of significant disadvantages. These disadvantages often increase the burden associated with receiving and applying patches. In some cases, this increased burden delays the application of some patches by some users, and even prevents the application of some patches by some users. Such delay and prevention in the application of patches can in some cases have serious negative consequences for users, especially for patches designed to reduce or eliminate a security risk.

[0007] One disadvantage of conventional approaches to patching relates to common cases in which multiple patches must be created in distributed to effect a single modification to a single program. In some cases, the program to be patched has several different "flavors" of a particular executable code module, such as a different flavor for each operating system or operating system version on which the program is designed to execute, and/or each natural language version of the program. Where the identified executable code module is such an executable code module, the patch creation and distribution process described above must be repeated for each flavor of the identified executable code module.

118

The user must then select and apply the patch for the appropriate flavor of the identified executable code module. Sorting through the resulting large number of patches and selecting the proper set of patches for application on each user's computer system can be very burdensome, so much so that this condition is sometimes called "patch hell." In some cases, an administrator must maintain an inventory database identifying the set of executable module versions installed on each target system, which is used to select appropriate conventional patches for each target system.

[0008] Another disadvantage of conventional approaches to patching relates to the large size of the distributed patches. It is not uncommon for executable code modules to have a size measured in megabytes, which in turn causes a single patch to have a comparable size, making it unwieldy to distribute and store, or even impossible to distribute and store, for some users. This problem can be multiplied for patches having multiple flavors. Further, because each conventional patch typically includes an entire substitute executable module, applying conventional patches can contribute to the problem of code churn.

[0009] A further disadvantage of conventional approaches to patching relates to a need by some users to test patches before applying them to production computer systems. In some cases, installing a patch on a computer system can have adverse results, such as where the new version of the identified executable code module contained in the patch introduces a new programming error, or where it causes a new, unpredicted interaction with another program running on the computer system against which it is applied. Accordingly, often before applying a patch against a production system whose data and operation are important to sustain, the user first applies the patch against a testing system to evaluate whether the patch is safe to apply to the production system. Such separate testing of patches adds to the burden associated with patching. Additionally, where a conventional patch creates a problem – such as an application compatibility problem or a new exploit vulnerability NOTARIA 18 DE BOGOTA D.C. ELIZABETH ... NOTARIA (E) 08 APR. 2005 ... BOTA FOTOCOPIA CON ... CON EL ORIGINAL ... TIENE A LA VISTA – substantially after the patch is applied, it can be difficult to trace such problems back to the patch.

[0010] An additional disadvantage of conventional approaches to patching relates to the operation of the installer included in the patch. Often, in order to replace an executable code module that is part of executing program, the installer must first terminate execution of that program. Also, in some cases, such replacement cannot be completed without restarting the computer system. Both of these steps can cause substantial disruptions in the use of the patched computer system.

[0011] Another disadvantage of conventional approaches to patching involves attempting to patch an executable module for which a "private fix," also called a "hot fix," has earlier been issued to a proper subset of the customers for that executable module. In such cases, because of difficulties encountered in distributing conventional patches that substitute different new versions of the executable code module depending upon to each user depending upon whether the user has applied the hot fix, it is typical to instead distribute a simple conventional patch that substitute a single new version of the executable module irrespective of whether the user has applied the hot fix. If that new version embodies the hot fix, the patch imposes the hot fix on customers not intended to receive it. If, on the other hand, that new version does not embody the hot fix, it deprives the customers intended to receive the hot fix of the hot fix.

[0012] Another disadvantage of conventional approaches to patching involves the fact that the installers for software products that rely on a particular executable module, such as a particular dynamic link library, often "hide" that executable module by storing it in a non-standard location in the target computer system's filesystem. Accordingly, it is sometimes difficult or impossible to determine whether a particular target system contains a copy of an executable module to be patched, and, if so, where it resides in the target computer system's filesystem. Also, some software products maintain a "catalog" of executable module versions that have been installed by their installers. A software product may rely upon the correctness of an indication in the catalog of the version of a particular executable module. Such reliance is defeated where a conventional patch replaces the

version of the executable module identified in the catalog with a new version of the executable module without updating the catalog.

[0013] Another disadvantage of conventional approaches to patching springs from the fact that they are impossible to apply at a time before the executable module to be patched is installed in the target computer system. As a result, if the executable module to be patched is installed in the target computer system after a conventional patch for that executable module is received, it is unlikely that the patch will be applied to the executable module.

[0014] Another disadvantage of conventional approaches to patching is that they typically can only be applied by a user logged in to the target computer system using an administrative account having liberal modification permissions. Logging into an administrative account for this purpose can make the target computer system vulnerable to viruses present on the target computer system seeking to modify aspects of the target computer system and requiring liberal permissions to do so.

[0015] Another disadvantage of conventional approaches to patching is that conventional patches can be difficult or impossible to disable, requiring steps such as reversing the replacement of an executable module, or reversing one or more modifications to the system registry.

[0016] Accordingly, a new approach to patching that overcame some or all of the disadvantages of conventional approaches to patching discussed above would have significant utility.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] Figure 1 illustrates an example of a suitable computing system environment in which the facility may be implemented.

[0018] Figure 2 is a data flow diagram showing a typical exchange of data between computer systems in accordance with the facility.

[0019] Figure 3 is a flow diagram showing steps typically performed by the facility in order to receive and process a new patch.

[0020] Figure 4 is a data structure diagram showing a sample patch table typical of those used by the facility.

[0021] Figure 5 is a flow diagram showing steps typically performed by the facility in order to update configuration instructions for a particular patch.

[0022] Figure 6 is a flow diagram showing steps typically performed by the facility to perform the parameter validation specified by a patch.

DETAILED DESCRIPTION

[0023] A software facility for patching installed computer program code ("the facility") is provided. In some embodiments, the facility adds parameter testing and test result handling to installed functions. In other embodiments, the facility adds various other kinds of functionality to installed functions, in some cases at arbitrary positions in the installed functions' flows of execution.

[0024] In some embodiments, for each patch, the facility distributes to each computer system to be patched – i.e., each "target computer system" – the specification of a point at which to perform a test, the identity of the test to perform, and how to act in response to one or more different test results. In some embodiments, the facility provides a standard set of parameter validation and other tests whose use can be specified in patches. For example, a patch may specify that, for a particular function, if a particular parameter of the function does not have a certain value, invocation of the function should fail before its substantive execution begins. Another patch may specify that, for a particular function, if a particular parameter has a length exceeding a specified maximum length, the parameter should be truncated to the specified maximum length before execution of the function is permitted to proceed. Many security exploits rely on causing functions to be called with parameter values that, while they were not blocked in the original version of a function's code, cause the function to create or take advantage of an unsafe condition. In many cases, such exploits can be prevented by using such patches to prevent the function from executing with such parameter values. In some embodiments, the patches specify that testing of

values other than function parameters, such as values read from a file or inputted by user.

[0025] In some embodiments, an automated patching agent automatically receives each patch, validates it, and stores it in a patch table for possible application. In some embodiments, each patch is applied to any instances of the executable module to be patched that are already loaded on the target computer system when the patch is received. This approach is referred to herein as "hot patching," and enables patches to become effective immediately upon being received, and does not require the facility to be able to determine where the executable module to be patched is stored on disk. In some embodiments, each received patch is applied to the disk image of the executable module to be patched, so that, when the disk image is loaded a future times, the loaded disk image includes the patch. This approach is referred to herein as "cold patching," and permits patches to be persistent across multiple sessions. In some embodiments, the facility performs both hot and cold patching. In some embodiments, each patch is applied to the executable module to be patched each time the executable module to be patched is loaded by the operating system's loader. This approach is referred to herein as "load-time patching." In some embodiments, each patch is applied to the executable module to be patched each time the function to be patched is invoked. This approach is referred to herein as "call-interception patching." Load-time patching call-interception patching both (1) do not require the facility to be able to determine where the executable module to be patched is stored on disk, (2) facilitate ready reversibility of a particular patch, and (3) do not require modification of the executable module's disk image.

[0026] In some embodiments, the facility permits a user or administrator to configure the operation of patches that have been applied. As examples, such configuration can include, for a particular applied patch: whether the test specified by the patch is performed when execution reaches the point specified for the patch; whether the test result handling specified by the patch is performed or ignored; and/or whether performance of the test and/or its result is logged,

displayed in warning message, etc. In these embodiments, the facility permits patches to be tested on production computer systems by initially enabling logging and disabling result handling. In these embodiments, the facility further permits operation of the patch to be logged in a "verbose mode" after enabling its result handling to assist with identifying instances in which the patch is creating a problem, such as an application compatibility problem or other IT problem. These embodiments also permit a patch to be quickly disabled after being applied if it is discovered that the patch is creating problems. Some embodiments of the facility also permit the patch to be quickly disabled by simply deleting the patch from the set of patches received and stored in the target computer system.

[0027] In some embodiments, the facility uses a "data-driven" patching approach, in which patches contain no code, but rather data, such as a small human-readable text or XML document, that specifies a point at which to perform a test, the identity of the test to perform, and how to act in response to one or more different test results. In such embodiments, a patching agent receives data-driven patches, and adds the tests and test handling specified by the patches. In some embodiments, the facility uses a "code-driven" patching approach, and which each patch contains a short program to be added to the executable module to be patched that itself performs the test by calling the facility's standard parameter testing functions, and that itself performs the test handling. Using data-driven patching or code-driven patching, it is sometimes possible to address all flavors of an executable module to be patched with a single patch.

[0028] In some embodiments, the facility signs each patch to demonstrate both that (1) the patch is from an approved source, and (2) the contents of the patch have not been modified since the time at which the patch was created by the approved source.

[0029] In some embodiments, the facility distributes every patch to every target computer system, and a patching agent on the target computer system determines automatically which patches or to be applied on the target computer system, and how they are to be applied, based upon the target computer system's

characteristics. This relieves users and administrators of many of the burdens conventionally associated with selecting and applying patches, and the burden of maintaining an accurate and current inventory database. For example, these characteristics can include which version of the executable module to be patched is installed on the target computer system. In these embodiments, the facility can overcome the type of problems typically caused by hot fixes, by distributing patches that specify different handling for hot-fixed and un-hot-fixed flavors of a particular executable module, obviating any need to either sacrifice a hot fix for a particular executable module or make the hot fix ubiquitous when patching that executable module.

[0030] In some embodiments, the patching agent stores every patch received in the target system, irrespective of whether the executable module to be patched by a particular patch is installed on the target system when that patch is received. Because the facility in many cases applies patches in response to the loading of an executable module to be patched or the invocation of a function to be patched, the facility can apply a patch to an executable module that was installed on the target system after that patch was received in the target system. Also, patches can survive the uninstallation and subsequent reinstallation of the executable module to be patched.

[0031] In some embodiments, the patching agent is implemented in an operating system service. In these embodiments, the facility accords the patching agent any permissions required to apply a patch. These embodiments reduce the security risk typically imposed when conventional patches are applied, as they obviate any need for a user to log in to the target computer system using an administrative account having broad modification permissions, and thereby give any viruses present on the target computer system a greater opportunity to modify sensitive aspects of the target computer system.

[0032] The patches used by the facility are typically relatively small, and therefore impose modest resource requirements for transmission and storage. Also, because the patches used by the facility tend to modify the behavior of the

patched software in a small number of well-defined ways, the facility helps to reduce the problem of code churn.

[0033] Figure 1 illustrates an example of a suitable computing system environment 100 in which the facility may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the facility. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0034] The facility is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the facility include, but are not limited to: personal computers, server computers, hand-held or laptop devices, tablet devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0035] The facility may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, and so forth, which perform particular tasks or implement particular abstract data types. The facility may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in local and/or remote computer storage media including memory storage devices.

[0036] With reference to Figure 1, an exemplary system for implementing the facility includes a general purpose computing device in the form of a computer 110. Components of the computer 110 may include, but are not limited to, a

processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0037] The computer 110 typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by the computer 110 and includes both volatile and nonvolatile media, and removable and non-removable media. By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can accessed by the computer 110. Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic,

RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer-readable media.

[0038] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Figure 1 illustrates operating system 134, application programs 135, other program modules 136 and program data 137.

[0039] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Figure 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0040] The drives and their associated computer storage media, discussed above and illustrated in Figure 1, provide storage of computer-readable instructions, data structures, program modules and other data for the computer 110. In Figure 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146 and program data 147.

Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers herein to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 110 through input devices such as a tablet, or electronic digitizer, 164, a microphone 163, a keyboard 162 and pointing device 161, commonly referred to as mouse, trackball or touch pad. Other input devices not shown in Figure 1 may include a joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. The monitor 191 may also be integrated with a touch-screen panel or the like. Note that the monitor and/or touch screen panel can be physically coupled to a housing in which the computing device 110 is incorporated, such as in a tablet-type personal computer. In addition, computers such as the computing device 110 may also include other peripheral output devices such as speakers 195 and printer 196, which may be connected through an output peripheral interface 194 or the like.

[0041] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer

networks, intranets and the Internet. For example, in the present facility, the computer system 110 may comprise source machine from which data is being migrated, and the remote computer 180 may comprise the destination machine. Note however that source and destination machines need not be connected by a network or any other means, but instead, data may be migrated via any media capable of being written by the source platform and read by the destination platform or platforms.

[0042] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160 or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Figure 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0043] While various functionalities and data are shown in Figure 1 as residing on particular computer systems that are arranged in a particular way, those skilled in the art will appreciate that such functionalities and data may be distributed in various other ways across computer systems in different arrangements. While computer systems configured as described above are typically used to support the operation of the facility, one of ordinary skill in the art will appreciate that the facility may be implemented using devices of various types and configurations, and having various components.

[0044] Figure 2 is a data flow diagram showing a typical exchange of data between computer systems in accordance with the facility. The computer systems

shown in Figure 2 (computer systems 210, 220, 221, 222, 230, 231, and 232) typically have some or all of the components shown and discussed in conjunction with Figure 1. At the patch distribution server, the facility generates one or more patches. These patches 201 are sent from the patch distribution server to one or more administration servers, such as administration servers 220 and 230. Each administration server, in turn, forwards the patches to one or more target computer systems, such as target computer systems 221, 222, 231, and 232. In some embodiments (not shown), the patch distribution server sends patches directly to one or more target computer systems, or via a more indirect route than through a single administration server. Patches received at a target computer system are processed at the target computer system as described in greater detail below. An administration server can also send patch configuration commands 202 to one or more target computer systems, which apply the patch configuration commands to reconfigure the operation of particular patches. As is discussed in greater detail below, a patch may be completely disabled; if a patch is not disabled, notification of its operation and its test result handling may each be independently enabled or disabled. When notification of its operation is enabled, notifications may be displayed or stored locally on the target computer system, or may be sent as notifications 203 to an appropriate administration server.

[0045] Figure 3 is a flow diagram showing steps typically performed by the facility in order to receive and process a new patch. In step 301, the facility receives a patch. The patch received in step 301 may be either a data-driven patch or a code-driven patch. A sample data-driven patch is shown in Table 1 below.

```
1       <Softpatch Patch="Q382429">
2        <AffectedApplication AffectedExe="sqlservr.exe">
3         <AffectedVersion Version="9.*">
4          <AffectedModules Name="SQLSORT.DLL">
5           <Version "8.0.*, 9.*">
6                <Function Name="SsrpEnumCore" Address="0x0802E76B"
7                    Param="2" Paramtype="LPSTR">
8                 <Filter MaxByteLength="60" />
9                 <Resolution ActionType="BOOL" Action="FALSE" />
10               </Function>
11          </Version>
12          <Version "10.*, 11.*">
13               <Function Name="SsrpEnumCore" Address="0x0802D283"
14                   Param="2" Paramtype="LPSTR">
15                <Filter MaxByteLength="128" />
16                <Resolution ActionType="BOOL" Action="FALSE" />
17               </Function>
18          </Version>
19         </AffectedModules>
20        </AffectedVersion>
21       </AffectedApplication>
22
23       <Signature Hash="MD5" Signature="C509-64AA-9161-8C52-
24        9F6D-BF5A-AEF2-ECE1-0038-34D1"/>
25      </Softpatch>
```

TABLE 1

[0046] Line 1 contains a unique identifier for the patch. Line 2 identifies the application affected by the patch. Line 3 identifies the application version affected by the patch. Line 4 identifies the executable module affected by the patch. Line 5 identifies two versions of the affected executable module – versions 8.0 .* and 9.* – for which patching directions are provided. Lines 6-10 contain patching directions for these two versions of the executable module. Lines 6-7 identify the function to be patched, its address in the executable module, its parameter to be tested by the patch, and the type of the parameter to be tested. Line 8 indicates that the parameter identified in lines 6-7 should be tested to determine whether its length exceeds 60 bytes. Line 9 indicates that the invocation of the function should fail if the test succeeds. Line 12 identifies two more versions of the affected executable module – versions 10.* and 11.*– for which patching directions are provided. Lines 13-17 contain patching directions for these two versions of the executable module. Lines 13-14 identify the function to be patched, its address in the executable module, its parameter to be tested by the

patch, and the type of the parameter to be tested. It can be seen that the address of the function to be patched in the executable module identified in lines 13-14 for versions 10 .*and 11 .* is different from the address of the function to be patched identified in lines 6-7 for versions 8.0 .* and 9.*. Line 15 indicates that the parameter identified in lines 13-14 should be tested to determine whether its length exceeds 128 bytes. Line 16 indicates that the invocation of the function should fail if the test succeeds. The patch may specify a variety of result handling action types, including failing the invocation of the patched function, raising an exception, terminating the process in which the patched executable module is being executed, or correcting the offending value (such as by truncating an overlong string). Lines 23-25 contain a signature for the patch that both identifies the source of the patch and verifies that the patch has not been altered since leaving its source.

[0047] Table 2 below contains a code-driven version of the patch shown in Table 1 above.

```
1     00411A7E  push     3Ch
2     00411A80  mov      eax,dword ptr [str]
3     00411A83  push     eax
4     00411A84  call     ValidateStringLength (411062h)
5     00411A89  add      esp,8
6     00411A8C  movzx    ecx,al
7     00411A8F  test     ecx,ecx
8     00411A91  je       411A9Ah
9     00411A93  jmp      foo+2 (411AD2h)
10    00411A9A  xor      eax,eax
11    00411A9C  ret
```

TABLE 2

[0048] Lines 1-3 push parameters for the testing function onto the stack. Line 4 calls the testing function. Lines 5-8 branch on the return code for the testing function. If the testing function succeeded, line 9 jumps back to begin executing the body of the patched function. If the testing function failed, lines 10-11 push a failure result code onto the stack and returns from the patched function to the color of the patched function. For readability, Table 2 omits certain details present in some code-driven patches, including machine signature, instructions

that test for the current values of the patch configuration flags, and instructions relocated from the beginning of the code for the patched function.

[0049] In some embodiments, patches of both types may contain additional information, including, for each of one or more versions of the executable module to be patched, a file signature that can be used to validate that a particular instance of the executable module is a proper copy of that version. Such a file signature may be, for example, a size or checksum for the entire executable module version, or the code expected to occur at a particular point in the executable module, such as at the offset at which the executable module is to be patched.

[0050] In step 302, if the patch is signed with a valid signature, then the facility continues in step 303, else the facility continues in step 301 to receive the next patch. In step 303, the facility adds the patch to a local patch table. In step 304, the facility initializes the initial configuration of the patch, such as by sending it to a default configuration.

[0051] Figure 4 is a data structure diagram showing a sample patch table typical of those used by the facility. The patch table 400 contains rows, such as rows 401 and 402, each divided into the following columns: a patch identifier column 411, containing the patch identifier extracted from the patch; an executable module column 412, containing information identifying the executable module to be patched, such as its name; an executable module versions column 413 identifying all of the versions of the executable module identified in column 412 to which the patch applies; a test performance enabled column 414, containing the current configuration value for whether the test specified by the patch should be performed each time the patched function is called; a test performance notification enabled column 415, containing the current configuration value for whether a notification should be generated each time the patch's test is performed; a test result notification enabled column 416, containing the current configuration value for whether a notification should be generated each time the patch's test succeeds; a test result handling enabled column 417, containing the current

configuration value for whether the patch's result handling should be implemented when the patch's test fails; and a patch column 418 that contains a pointer to the patch itself, specifying a test and test result handling to be performed each time the test fails. In some embodiments, rather than containing a pointer to the patch as shown, the patch column 418 contains each patch directly. A particular patch table may contain or point to patches of various types, such as all code-driven patches, all data-driven patches, or a combination of code-driven and data-driven patches.

[0052] In step 305, once the facility has added the received patch to the patch table and initialized its configurations, the patch is available to be automatically applied by the facility to the executable module. In step 305, the facility may utilize a variety of approaches to applying the patch, including those described in the application incorporated by reference, as well as real-time function call interception, and/or code rewriting of (1) already-loaded executable modules, (2) one or more disk images of the executable module, or (3) instances of executable modules loaded by the operating system's loader. After step 305, the facility continues in step 301 to receive the next patch.

[0053] Figure 5 is a flow diagram showing steps typically performed by the facility in order to update configuration instructions for a particular patch. In step 501, the facility receives configuration instructions for a particular patch, such as from an administrator. In some embodiments, such configuration instructions may be generated by administrators using group policies. In step 502, the facility updates the patch is configuration in the patch table in accordance with the received instructions. After step 502, the facility continues in step 501 to receive the next configuration instructions.

[0054] Figure 6 is a flow diagram showing steps typically performed by the facility to perform the parameter validation specified by a patch. In step 601, a patched function is called. In step 602, if testing is enabled for the patch that affects the called function, then the facility continues in step 603, else the facility continues in step 601 to process the next call to a patched function. In step 603, if test

performance notification is enabled for the patch, then the facility continues in step 604, else the facility continues in step 605. In step 604, the facility generates a notification that the test was performed. Steps 604, 608, and 610 may involve displaying or storing an indication on the target computer system that the test was satisfied, and/or transmitting such an indication to a remote computer system for display or logging there.

[0055] In step 605, the facility performs the validation test specified by the patch. In some embodiments, step 605 involves calling one of a group of standard routines utilized by the facility for testing. In step 606, if the test performed in step 605 is satisfied, then the facility continues in step 601, else the facility continues in step 607. In step 607, if test result notification is enabled for the patch, then the facility continues in step 608, else the facility continues in step 609. In step 608, the facility generates a notification that the test was not satisfied. In step 609, if test result-handling is enabled for the patch, then the facility continues in step 610, else the facility continues in step 601. In step 610, the facility performs the test result handling specified by the patch. After step 610, the facility continues in step 601.

[0056] It will be appreciated by those skilled in the art that the above-described facility may be straightforwardly adapted or extended in various ways. For example, the facility may be used to apply a variety of different kinds of patches in a variety of ways at a variety of locations in executable modules of a variety of types for a variety of purposes. Also, while patches are described herein as containing value validation tests that indicate a problem when they fail, the facility may also be implemented using value invalidity tests that indicate a problem when they succeed. In some embodiments, each test is accompanied by an indication of whether its success or failure indicates a problem. While the foregoing description makes reference to preferred embodiments, the scope of the invention is defined solely by the claims that follow and the elements recited therein.

CLAIMS

We claim:

[c1] 1. A method in a computing system for augmenting software, comprising:

receiving in a target computer system an augmentation specification specifying (a) a function to be augmented, the specified function being among software executed in the computing system, (b) a parameter of the function to be tested, (c) a test to apply to the specified parameter, and (d) a modification to perform to the behavior of the function if the specified test is not satisfied by the specified parameter; and

when the specified function is invoked on the target computer system, if the specified test is not satisfied by the specified parameter, performing the specified modification to the behavior of the specified function.

[c2] 2. The method of claim 1 wherein the modification specified by the augmentation specification is preventing the execution of the specified function.

[c3] 3. The method of claim 1 modification specified by the augmentation specification is executing the specified function after alteration of the specified parameter.

[c4] 4. The method of claim 1, wherein a plurality of augmentation specifications are received in the target computer system.

[c5] 5. The method of claim 4, further comprising maintaining all of the received augmentation specifications in a data structure.

[c6] 6. The method of claim 1 wherein no user intervention is required subsequent to receiving the augmentation specification in order to perform the specified modification to the behavior of the specified function.

[c7] 7. The method of claim 1 wherein the augmentation specification specifies a test to applied to the specified parameter by identifying a parameter testing function to invoke whose code is not included in the augmentation specification.

[c8] 8. The method of claim 1 wherein the augmentation specification specifies a test to be applied to the specified parameter by identifying a parameter testing function to invoke that was installed before the augmentation specification was received.

[c9] 9. The method of claim 1 wherein the augmentation specification contains code.

[c10] 10. The method of claim 9 wherein the code contained by the augmentation specification includes code invoking a parameter testing function whose code is not included in the augmentation specification.

[c11] 11. The method of claim 1 wherein the augmentation specification contains text identifying a parameter testing function whose code is not included in the augmentation specification.

[c12] 12. The method of claim 11 wherein the code of the parameter testing function identified by the text contained by the augmentation specification is contained by the augmentation specification.

[c13] 13. The method of claim 11 wherein the code of the parameter testing function identified by the text not contained by the augmentation specification is not contained by the augmentation specification.

[14. The method of claim 1 further comprising providing an interface for configuring a state for controlling the operation of the augmentation specification, and wherein the specified modification to the behavior of the specified function is performed only when the state is an enabled state.

[c15] 15. The method of claim 14 wherein the provided interface enables the operation of the augmentation specification to be configured locally.

[c16] 16. The method of claim 14 wherein the provided interface enables the operation of the augmentation specification to be configured remotely.

[c17] 17. The method of claim 14 wherein the provided interface enables the operation of the augmentation specification to be configured in accordance with a policy.

[c18] 18. The method of claim 14 wherein the specified test is performed before any substantive code of the specified function is executed.

[c19] 19. The method of claim 1 further comprising providing an interface for configuring warnings associated with the augmentation specification, and wherein, if the provided interface is used to enable warnings, generating a warning each time the test specified by the augmentation specification fails.

[c20] 20. The method of claim 1 further comprising providing an interface for configuring notifications associated with the augmentation

specification, and wherein, if the provided interface is used to enable notifications, generating a notification each time the test specified by the augmentation specification is performed.

[c21] 21. The method of claim 20, further comprising consolidating the generated notifications into a single consolidated notification.

[c22] 22. The method of claim 1 wherein the received augmentation specification is signed,
the method further comprising verifying the signature of the augmentation specification,
and wherein the specified modification to the behavior of the specified function is performed only if verification of the signature of the augmentation specification was successful.

[c23] 23. The method of claim 22 wherein the specified modification to the behavior of the specified function is performed only if the signer of the signature of the augmentation specification matches a signer of the software containing the specified function.

[c24] 24. The method of claim 1, further comprising performing the specified test in response to detecting that the specified function has been invoked, without altering the code of the specified function.

[c25] 25. The method of claim 1, further comprising altering the code of the specified function to perform the specified test when the specified function is invoked.

[c26] 26. The method of claim 25 wherein the code of the specified function is altered by inserting into the code of the specified function a jump to a separate routine that performs the specified test.

[c27] 27. The method of claim 1 wherein the specified function is provided in a distinguished executable module that is loadable using a loader, the method further comprising registering with the loader to have a routine that performs the specified test called before the specified function is executed in response to each invocation of the specified function.

[c28] 28. A computing system that enables augmentation of software present in the computing system, comprising:

a storage device containing software;

a patch receiver that receives in the computing system a patch specifying (a) a point at which the software is to be augmented, (b) a value associated with the function to be tested at the specified point, (c) a test to apply to the specified value, and (d) a modification to perform to the behavior of the software if the specified test is not satisfied by the specified value; and

a patching agent that injects code into the software at the specified point such that, when execution of the software reaches the specified point on the target computer system, if the specified test is not satisfied by the specified value, the specified modification to the behavior of the software is performed.

[c29] 29. A computer-readable medium whose contents cause a target computing system to perform a method for adding value validation to software available in the target computing system, comprising:

receiving in the computing system an augmentation specification specifying (a) a function to which value validation is to be added, the specified function being among software available in the computing system, (b) data to be tested that exists in the target com[illegible] execution of the function,

(c) a test to apply to the specified data, and (d) a modification to perform to the behavior of the function if the specified test is not satisfied by the specified data; and

when the specified function is invoked on the target computer system, if the specified test is *not* satisfied by the specified data, performing the specified modification to the behavior of the specified function.

[c30] 30. The computer-readable medium of claim 22 wherein the specified test is a test that is always satisfied, irrespective of the value of the specified data.

[c31] 31. A generated data signal conveying a patch data structure, comprising:

information identifying a function to which value validation is to be added;

information identifying data to be tested that exists during execution of the function;

information identifying a test to apply to the specified data; and to

information identifying a modification to perform to the behavior of the function if the specified test is not satisfied by the specified data,

such that, if the data signal is received on a target computer system on which the specified function is executed, the contents of the data structure may be used to perform the specified modification to the behavior of the specified function if the specified test is not satisfied by the specified data when the specified function is invoked on the target computer system.

[c32] 32. The generated data signal of claim 31 wherein the generated data signal is transmitted between computer systems.

[c33] 33. The generated data signal of claim 31 wherein the generated data signal is transmitted within a computer system.

[c34] 34. A method for adding value validation to software available in the target computing system, comprising:

receiving in the computing system an augmentation specification specifying (a) software to which value validation is to be added, (b) a point in the specified software at which value validation is to be added, (c) data to be tested that exists in the target computing system during execution of the specified software at the specified point, (d) a test to apply to the specified data, and (e) a modification to perform to the behavior of the software if the specified test is not satisfied by the specified data;

enabling a user to configure an operational mode for the received augmentation specification; and

when the specified software is executed at the specified point on the target computer system, applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification.

[c35] 35. The method of claim 34 wherein the user configures an active operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises, if the specified test is not satisfied by the specified data, performing the specified modification to the behavior of the specified software.

[c36] 36. The method of claim 34 wherein the user configures an active diagnostic operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification

143

comprises, if the specified test is not satisfied by the specified data, both (1) performing the specified modification to the behavior of the specified software, and (2) generating a notification that the specified test is not satisfied by the specified data.

[c37] 37. The method of claim 34 wherein the user configures a verbose active operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises:

if the specified test is not satisfied by the specified data, performing the specified modification to the behavior of the specified software; and

generating a notification that the specified test was performed.

[c38] 38. The method of claim 34 wherein the user configures an active verbose diagnostic operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises:

if the specified test is not satisfied by the specified data, both (1) performing the specified modification to the behavior of the specified software, and (2) generating a notification that the specified test is not satisfied by the specified data; and

generating a notification that the specified test was performed.

[c39] 39. The method of claim 34 wherein the user configures an inactive operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises omitting to perform the specified modification to the behavior of the

specified software, irrespective of whether the specified test is satisfied by the specified data.

[c40] 40. The method of claim 34 wherein the user configures an inactive diagnostic operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises:

omitting to perform the specified modification to the behavior of the specified software, irrespective of whether the specified test is satisfied by the specified data; and

if the specified test is not satisfied by the specified data, generating a notification that the specified test is not satisfied by the specified data.

[c41] 41. The method of claim 34 wherein the user configures an inactive diagnostic operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises:

omitting to perform the specified modification to the behavior of the specified software, irrespective of whether the specified test is satisfied by the specified data; and

generating a notification that the specified test was performed.

[c42] 42. The method of claim 34 wherein the user configures an inactive diagnostic operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises:

omitting to perform the specified modification to the behavior of the specified software, irrespective of whether the specified test is satisfied by the specified data;

generating a notification that the specified test was performed; and

if the specified test is not satisfied by the specified data, generating a notification that the specified test is not satisfied by the specified data.

EFFICIENT PATCHING

ABSTRACT

A facility for augmenting software in a target computer system is described. The facility receives and augmentation specification in the target computer system. The augmentations specification specifies: (a) a function to be augmented, (b) a parameter of the function to be tested, (c) a test to apply to the specified parameter, and (d) and modification to perform to the behavior of the function if the specified test is not satisfied by the specified parameter. When the specified function is invoked on the target computer system, if the specified tested is not satisfied by the specified parameter, the facility performs the specified modification to the behavior of the specified function.

148

EXPRESS MAIL NO. EV336672580US

PATENT APPLICATION

DECLARATION AND POWER OF ATTORNEY

ATTORNEY DOCKET NO. 311826.01/USP1 MS DOCKET NO. 305697.2b

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: RESIDENT PATCHING

the specification of which is filed herewith unless the following box is checked:

( ) was filed on ____ _____ as US Application Serial No. or PCT International Application Number _____ and was amended on _____ _____ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35 U.S.C. 119
			YES___ NO
			YES___ NO

POWER OF ATTORNEY

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) associated with

Customer No. 25096
P.O. Box 1247
Seattle, Washington 98111-1247

to prosecute this application and transact all business in the Patent and Trademark Office connected therewith: STEPHEN E. ARNETT, Registration No 47,392; ROGER K. CARREYN, Registration No. 33,774; MAY Y. CHAN, Registration No. 51,085; BRIAN R. CYLKMAN, Registration No. 38,165; CHRISTOPHER DALEY-WATSON, Registration No. 34,807; PETER J. DEHLINGER, Registration No. 39,026; DAVID DEVLIN, Registration No. 48,870; DAVID BOGART DOFT, Registration No. 30,228; DAVID T. DUTCHER, Registration No. 51,488; LEMANN GOETHEY, Registration No. 37,197; JOSEPH HAMILTON, Registration No. 51,778; PAUL L. HICKMAN, Registration No. 28,516; EDWARD S. HOTCHKISS, Registration No. 35,909; STEVEN KELLEY, Registration No. 43,449; EU TE KIM, Registration No. 46,231; JONATHAN F. KLEIN A, Registration No. 47,726; STEVEN D. LAWRENZ, Registration No. 37,376; JACQUELINE P. MAHONEY, Registration No. 48,597; SHAILESH MEHTA, Registration No. 44,804; JUDY M. MOHR, Registration No. 34,784; CHUN M. NG, Registration No. 36,878; KENNETH H. OHRINER, Registration No. 31,644; PAUL T. PARKER, Registration No. 38,264; MAURICE J. PIRIO, Registration No. 33,273; AARON J. POLEDNA, Registration No. 54,675; RAJIV P. RADHY, Registration No. 38,592; MICHELLE SAQUE, Registration No. 50,626; TIM R. SEELEY, Registration No. 33,575; LAUREN SLICER, Registration No. 51,564; CARINA M. TAN, Registration No. 45,965; LANEY W. THROWER, Registration No. 47,994; GLENN P. VON TRESCH, Registration No. 45,264; JOHN M. WICKHAM, Registration No. 42,216; MICHAEL J. WISE, Registration No. 54,067; ROBERT G. WOULSTON, Registration No. 37,282; JAMES J. ZHU, Registration No. 52,995; all affiliated with Perkins Coie LLP, along with PATRICIA E. BONNER, Registration No. 37,086; DAVID BARTLEY EFFINAUER, Reg. No. 38,693; STACY QUAN, Registration No. 53,790; JEFFREY L. RANCK, Registration No. 38,592; MARTIN L. SHIVELY, Registration No. 53,552; and JOHN WERESH, Registration No. 53,592; of Microsoft Corporation, One Microsoft Way, Redmond, Washington 98052 as the principal attorneys with full power of substitution, association, and revocation to prosecute said application, to transact all business in the Patent and Trademark Office connected therewith, and to receive the letters patent thereon.

Send Correspondence to:	Direct Telephone Calls To:
Contact Name	Steven D. Lawrenz
Firm Name	Perkins Coie LLP
Firm Address	P.O. Box 1247
City, State, and Zip	Seattle, WA 98111-1247
	Contact Phone No. (509) 359-8000

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 ABR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL QUE TUVO A LA VISTA

148

149

5

DECLARATION AND POWER OF ATTORNEY

ATTORNEY DOCKET NO. [illegible] MS DOCKET NO. [illegible]

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Inventor: Anthony Mumfield Citizenship: British

Residence: Redmond, Washington

Post Office Address: 18229 NE 28th Street, Redmond, Washington 98052

Anthony Mumfield — Date: 1/8/03

Full Name of Inventor: Gilad Golan Citizenship: Israel

Residence: Redmond, Washington

Post Office Address: 16908 NE 45th Ct., Redmond, Washington 98052

Gilad Golan — Date

Full Name of Inventor: Jason Garms Citizenship: US

Residence: Woodinville, Washington

Post Office Address: 12816 199th Ct. NE, Woodinville, Washington 98077

Jason Garms — Date

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 ABR 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

149

150

DECLARATION AND POWER OF ATTORNEY

ATTORNEY DOCKET NO. [illegible] MS DOCKET NO. [illegible]

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Inventor: Anthony Blumfield Citizenship: British

Residence: Redmond, Washington

Post Office Address: 18839 NE 29th Street, Redmond, Washington 98052

______________________ ______________________

Anthony Blumfield Date

Full Name of Inventor: Gilad Golan Citizenship: Israel

Residence: Redmond, Washington

Post Office Address: 16906 NE 45th Ct., Redmond, Washington 98052

______________________ 6/4/04

Gilad Golan Date

Full Name of Inventor: Jason Garms Citizenship: US

Residence: Woodinville, Washington

Post Office Address: 12014 190th Ct. NE, Woodinville, Washington 98077

______________________ ______________________

Jason Garms Date

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 ABR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

TDQ

151

DECLARATION AND POWER OF ATTORNEY

ATTORNEY DOCKET NO. [illegible] MS DOCKET NO. [illegible]

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Inventor: Anthony Rhodefield Citizenship: British

Residence: Redmond, Washington

Post Office Address: 18225 NE 59th Street, Redmond, Washington 98052

Anthony Rhodefield Date

Full Name of Inventor: Oliad Golan Citizenship: Israel

Residence: Redmond, Washington

Post Office Address: 16926 NE 69th Ct., Redmond, Washington 98052

Oliad Golan Date

Full Name of Inventor: Jason Garms Citizenship: US

Residence: Woodinville, Washington

Post Office Address: 13854 190th Ct. NE, Woodinville, Washington 98077

X [signature] 24 June 2004

Date

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 ABR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

151

152

Full Name of Inventor: Raed Abdelhaq Citizenship: US

Residence: Issaquah, Washington

Post Office Address: [illegible]

Raed Abdelhaq

6-4-04
Date

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
08 ABR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL ORIGINAL QUE TUVO A LA VISTA

152

153

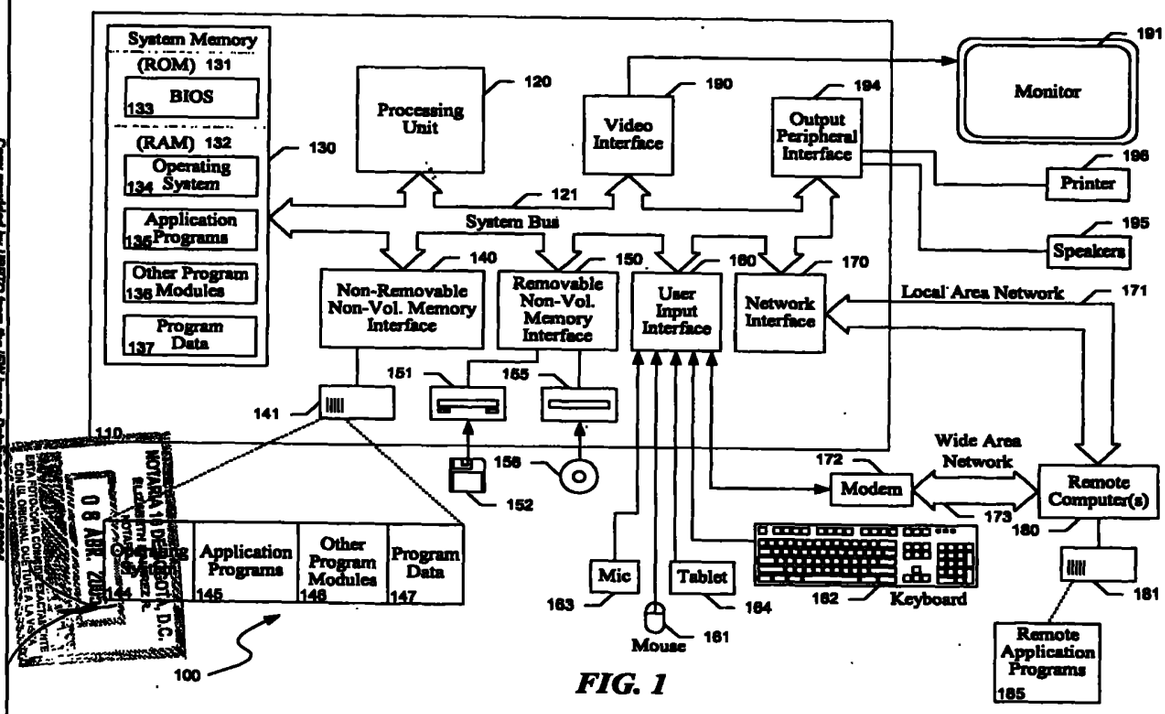


FIG. 1

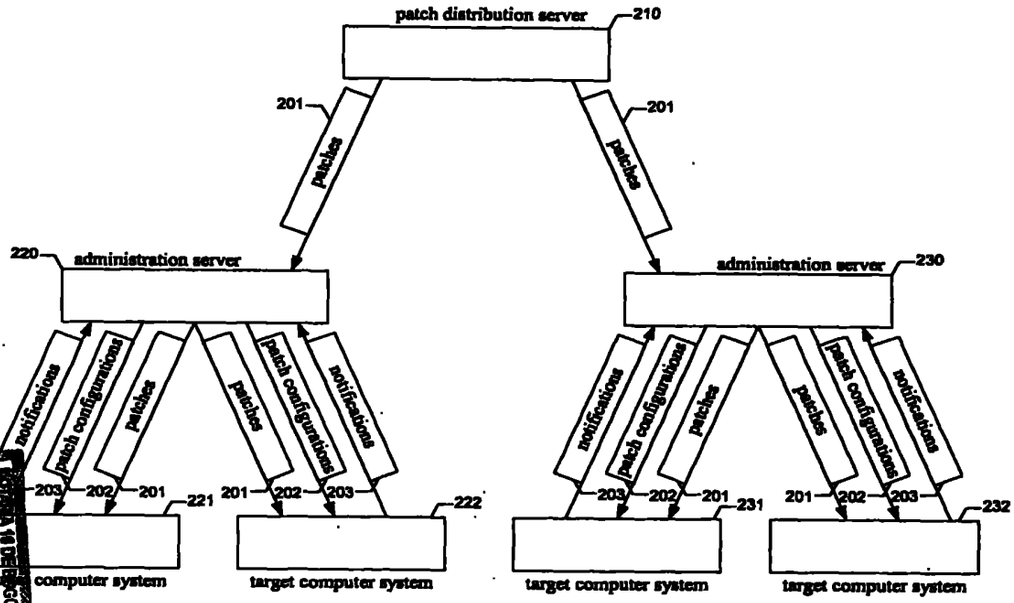


FIG. 2

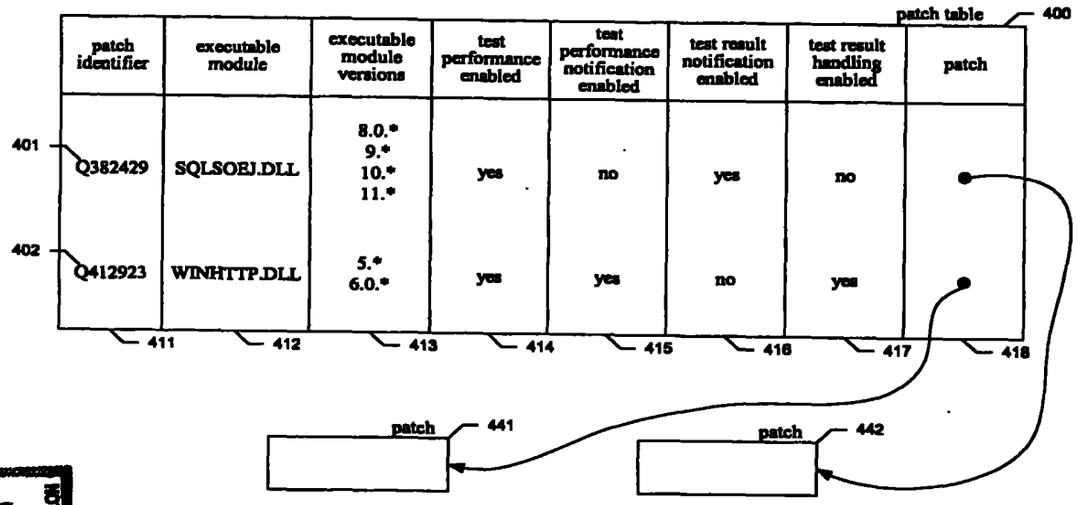


FIG. 4

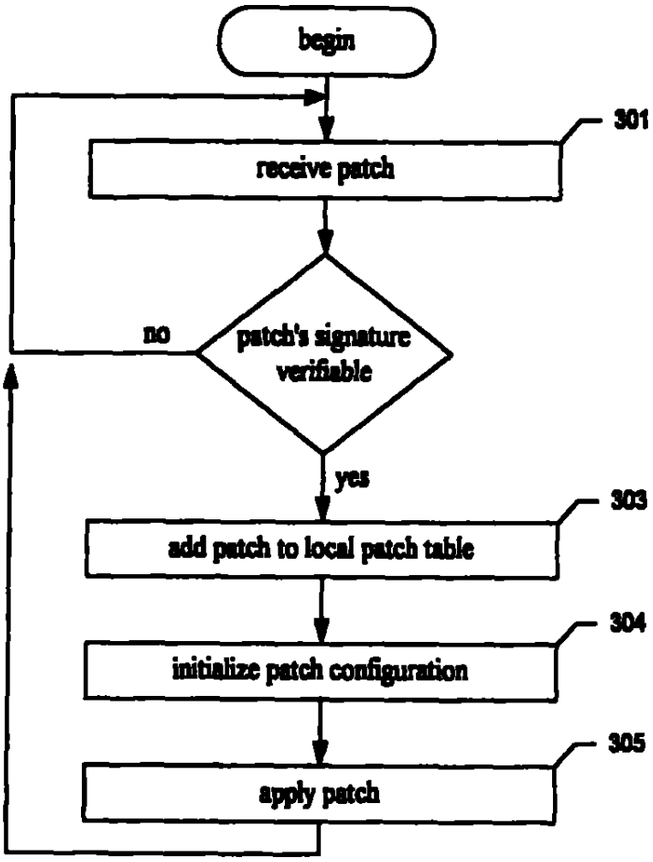


FIG. 3

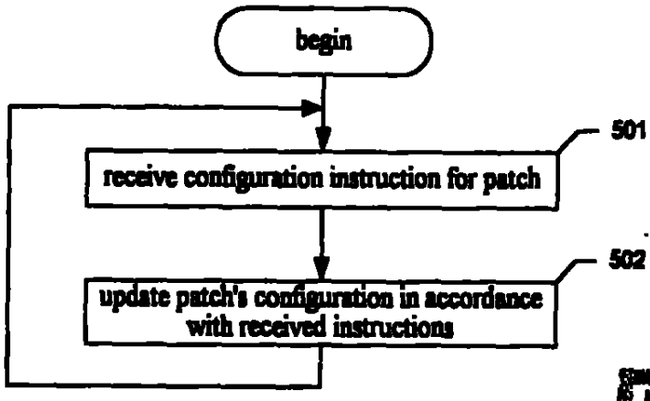


FIG. 5

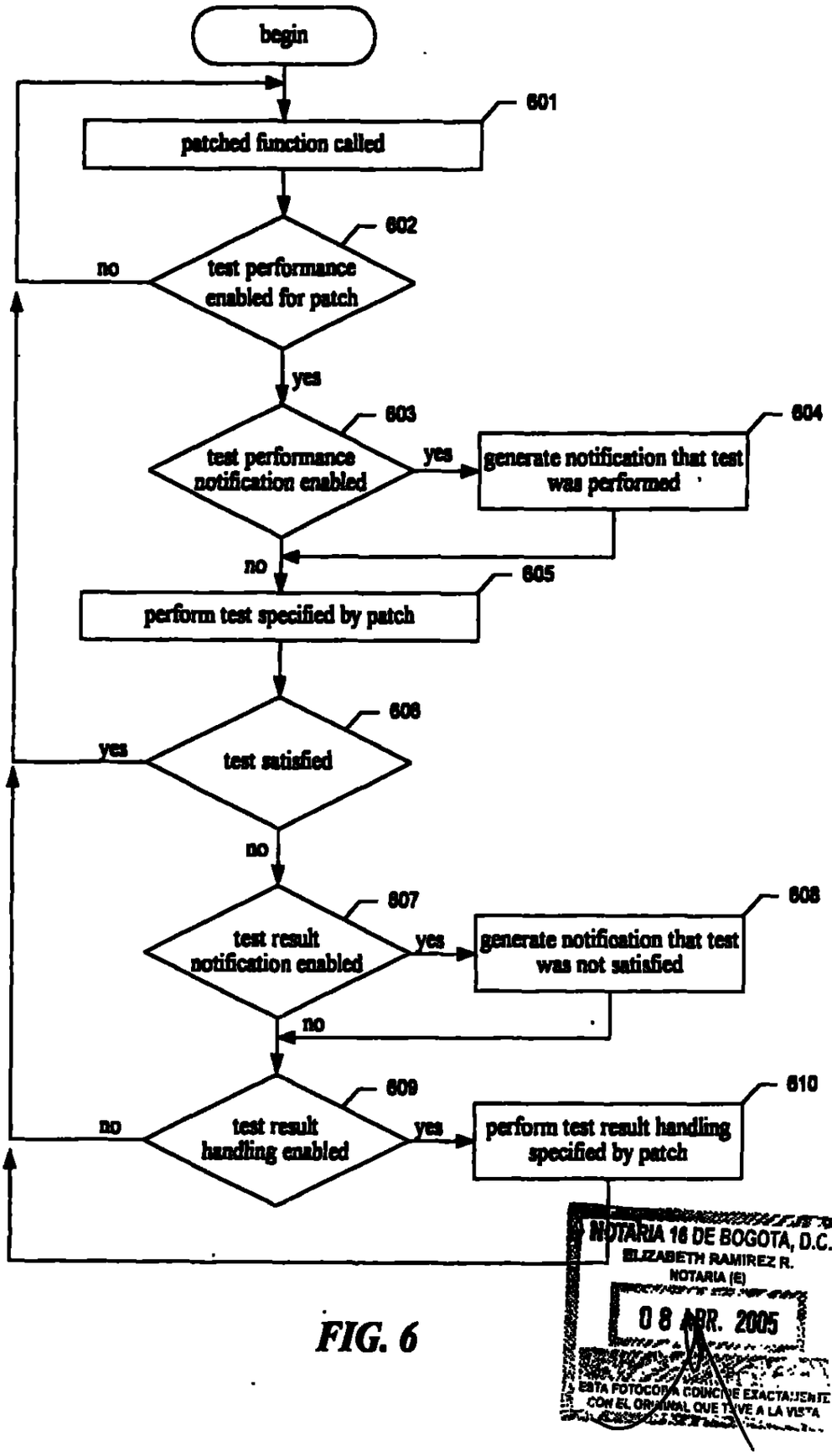


FIG. 6

158 2

TRADUCCIÓN PARCIAL DEL INGLÉS AL ESPAÑOL DE UN DOCUMENTO ORIGINAL ENTREGADO EN UN FOLDER DE LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DEL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS, CON COBERTURA DE PLÁSTICO TRANSPARENTE, REALIZADA POR MERYAM ADAM, TRADUCTORA E INTÉRPRETE OFICIAL.

PRIMERA PÁGINA: Emitida en papelería especial con reborde negro-gris con el número PA 1243337 en la parte superior izquierda del documento: **ESTADOS UNIDOS DE AMÉRICA - A TODOS LOS BENEFICIARIOS DE ESTE DOCUMENTO: EL DEPARTAMENTO DE COMERCIO DE LOS ESTADOS UNIDOS** Oficina de Patentes y Marcas Registradas de los Estados Unidos

Noviembre 02 de 2004

ESTE DOCUMENTO CERTIFICA QUE LOS ANEXOS SON COPIA FIEL DE LOS D CUMENTOS REGISTRADOS EN LA OFICINA DE PATENTES Y MARCAS REGISTRADAS DE LOS ESTADOS UNIDOS CORRESPONDIENTES A LA S LICITUD DE PATENTE IDENTIFICADA A CONTINUACIÓN, QUE CUMPLEN CON LOS REQUISITOS PARA QUE LES SEA OTORGADA UNA FECHA DE REGISTRO DE ACUERDO CON 35 USC 111.

NÚMERO DE SOLICITUD: ***60/570,124***

FECHA DE REGISTRO: ***11 de mayo de 2004***

Estampilla dorada del Departamento de Comercio - Oficina de Patentes y Marcas Registradas de los Estados Unidos

Bajo la potestad del COMISIONADO DE PATENTES Y MARCAS REGISTRADAS

(Firma)

L. EDELEN

Funcionario de Certificaciones

Estampilla dorada con emblema en el centro y a su alrededor "Oficina de Patentes y Marcas Registradas de los Estados Unidos - Departamento de Comercio"

NOTARIA 16 DE BOGOTA, D.C.
ELIZABETH RAMIREZ R.
NOTARIA (E)
2005
ESTA FOTOCOPIA CORRESPONDE FIELMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

158

159

PTO/SB/16 (5-03)
Aprobado para uso hasta el 07/31/2006 OMB 0651-0032
Oficina de Patentes y Marcas de los Estados Unidos, Departamento de Comercio de los Estados Unidos.
Bajo la reconocibilidad de reducción de papel de 1995. Las personas no están requeridas para contestar a la información a menos que esta sea distribuida válida por un número de control de OMB.

CUBIERTA PARA SOLICITUD PROVISIONAL DE PATENTE

Este es un requisito para presentar una solicitud provisional para patente bajo el 37 CFR 1.53 (c)

ETIQUETA DE CORREO EXPRESO Nº EV336675073US

INVENTOR(ES)

Nombre dado (primero y medio (si hay))	Apellidos	Residencia (ciudad y estado o país extranjero)

☐ Inventores adicionales son nombrados en _____ hojas anexas separadamente numeradas

TITULO DEL INVENTO (500 caracteres max)

PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE

Dirigir toda la correspondencia a: **DIRECCION DE CORRESPONDENCIA**

☒ Número de cliente 25096 ⟶
O

☒ Firma o nombre individual	PERKINS COLE LLP

PARTES DE LA SOLICITUD ADJUNTAS (Marque todo lo que aplique)

☒ Especificación Numero de páginas 49 ☐ CD(s) Número

X Dibujos Numero de páginas 5 x Otro (especificar) Tarjeta postal

☐ Hoja de Solicitud de datos Ver 37 CFR 1.76

METODO DE PAGO PARA LAS TASAS DE ESTA SOLICITUD PROVISIONAL PARA PATENTE (Marque uno)

VALOR DE LA TASA ($)

O El Solicitante reivindica su estado como pequeña entidad
O Se anexa un cheque o giro postal para cubrir las tasas de Presentación
O El director se encuentra autorizado para cargar las tasas o créditos
O cualquier pago en exceso a la Cuenta de depósito número 50-0665
O Pago con tarjeta de crédito. Se anexa Formato PTO-2038.

El Invento fue hecho por una agencia del Gobierno de los Estados Unidos o bajo contrato con una agencia del Gobierno de los Estados Unidos

☒ No

☐ Si, el nombre de la Agencia del Gobierno de los E.U. y el número de contrato de Gobierno es: _____

Respetuosamente presentado,
FIRMA (firma legible)
NOMBRE DIGITADO O IMPRESO STEVEN D. LAWRENZ
TELEFONO (206) 359-8000

Fecha 11 de mayo de 2004
REGISTRO No. 37,376
(si es apropiado)
Expediente No. 41826.8010US

USO EXCLUSIVO PARA PRESENTAR UNA SOLICITUD PROVISIONAL DE PATENTE

Esta colección de información es requerida por 37 CFR 1.51. La información en este formato puede ser pública para archivar (y por el PTO para procesar) una solicitud provisional. La confidencialidad es regulada por el 35 U.S.C. 122 y 37 CFR 1.14. Esta colección toma 8 horas para ser completadas incluyendo recolección, preparación y presentación de la solicitud provisional al PTO. El tiempo puede variar según cada caso individual. Cualquier comentario sobre la cantidad de tiempo que usted requirió por llenar esta forma y/o sugerencias para reducir esta obligación deben ser enviado al oficial jefe de Información de la Oficina de Marcas y Patentes de los E.U. Departamento de Comercio de los E.U. P.O. Box 1450 Alexandria VA 22313-1450. NO ENVIE PAGOS O FORMATOS COMPLETOS A ESTA DIRECCION. ENVÍELOS A: Parada de correo solicitud provisional, Comisionado para Patentes, P.O. Box 1450 Alexandria VA 22313-1450

Copia proporcionada por USPTO de la base de datos de imagen PACR el 10-29-2007

Nota del Traductor: Esta página se encuentra sellada en su parte superior izquierda con Número 17712 U.S. PTO/código de barras No.051104; y en su parte superior derecha 22264 U.S. PTO 60/570124 código de barras No. 051104.

159

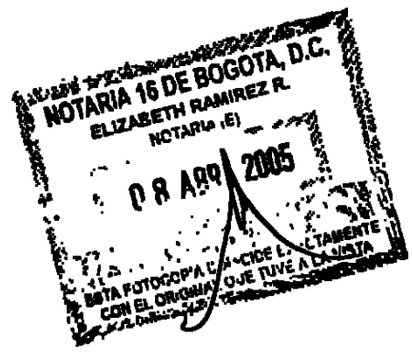
160

CORREO EXPRESO No. EV336675073US

PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE

REFERENCIA CRUZADA A SOLICITUDES RELACIONADAS

[0001] Esta solicitud está relacionada con la Solicitud de Patente de los Estados Unidos No. 10/307,902 titulada "PARCHEO EN FUNCIONES DE USO EN UN SISTEMA DE COMPUTO EN USO" y presentada en Diciembre 2, 2002; solicitud de patente de los Estados Unidos No. _____ (numero de archivo del abogado interno _____) titulada "PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE", presentada concurrentemente con la presente, cada una de los cuales está siendo incorporado por referencia a la presente en su integridad



160

EL CAMPO TÉCNICO

[0001] La invención presente está dirigida al campo de actualización de la operación de programas de computadora instalados.

EL FONDO

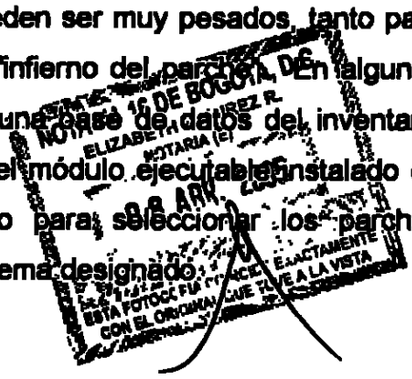
[0002] "Reparchar" o remendar es el proceso de modificar los programas ya-instalados, incluso los programas de aplicación, programas utilitarios, sistemas operativos y componentes del sistema operativo, controladores de dispositivos, etc. Remendar puede ser útil para modificar los programas de una variedad de propósitos, incluyendo la corrección de un error de programación, reduciendo o eliminando un riesgo de seguridad, o mejorando la lógica usada por el programa modificado. El "Reparcheo" es iniciado típicamente por la compañía u otra organización que originalmente proporcionó el programa a ser remendado.

[0003] Los programas instalados están predominantemente compuestos de módulos de código ejecutables. Como ejemplo, muchos programas diseñados para ejecutarse sobre el sistema operativo Windows XP de Microsoft Corp. de Redmond, Washington están predominantemente compuestos de módulos de código ejecutables llamados "DLLs." Un acercamiento convencional popular para remendar es identificar, entre los módulos de código ejecutables que constituyen el programa instalado a ser remendado, el módulo de código ejecutable que contiene el código del programa que uno desea modificar con un parche; creando una nueva versión del módulo de código ejecutable identificado en el cual se desea que modificación sea hecha; y distribuyendo la nueva versión del módulo de código ejecutable identificado, junto con un programa instalador, a usuarios que pueden desear aplicar el parche. Cada usuario determina entonces si desea aplicar el parche, y en ese caso, ejecuta el programa instalador que

reemplaza la versión original del módulo del código ejecutable identificado con la nueva versión del módulo del código ejecutable identificado.

[0004] Los acercamientos convencionales para remendar tienen varios desventajas significantes. Estas desventajas aumentan a menudo la carga asociada con recibir y aplicar los parches. En algunos casos, este aumento de carga tarda la aplicación de algunos parches por algunos usuarios, e incluso previene la aplicación de algunos parches por algunos usuarios. Tal retraso y prevención en la aplicación de parches pueden en algunos casos tener consecuencias negativas serias para los usuarios, sobre todo para los parches diseñados para reducir o eliminar un riesgo de seguridad.

[0005] Una desventaja de acercamientos convencionales a remendar relacionada a casos comunes en que los parches múltiples deben crearse distribuidos para efectuar una sola modificación a un solo programa. En algunos casos, el programa a ser remendado tiene varios "sabores" diferentes de un módulo de código ejecutable particular, como un sabor diferente para cada sistema operativo o versión del sistema operativo en que el programa esta diseñado para ejecutarse, y/o cada versión del idioma natural del programa. Donde el módulo de código ejecutable identificado es tal un módulo de código ejecutable, la creación del parche y el proceso de distribución descritos para cada sabor del módulo de código ejecutable identificado. El usuario debe seleccionar entonces y debe aplicar el parche para el sabor apropiado del módulo de código ejecutable identificado. El ordenamiento a través del número grande resultante de parches y seleccionar el conjunto apropiado de parches para la aplicación en el sistema de cómputo de cada usuario pueden ser muy pesados, tanto para que esta condición a veces se llame el "infierno del parche". En algunos casos, un administrador debe mantener una base de datos del inventario que identifica el conjunto de versiones del módulo ejecutable instalado en cada sistema designado que es usado para seleccionar los parches convencionales apropiados para cada sistema designado.



116

[0006] Otra desventaja de acercamientos convencionales para remendar se relaciona al tamaño grande de los parches distribuidos. No es raro para los módulos de código ejecutables tener un tamaño medido en megabytes que a su vez causa que un solo parche tenga ese tamaño comparable, haciéndolo pesado para distribuir y guardar, o incluso imposible de distribuir y guardar, para algunos usuarios. Este problema puede multiplicarse para parches que tienen múltiples sabores. Más allá, porque cada parche convencional incluye típicamente a un suplente completo del módulo ejecutable, mientras se aplican los parches convencionales pueden contribuir al problema de errores de código.

[0007] Una desventaja adicional al acercamiento convencional de remendar se relaciona a una necesidad para algunos usuarios de probar los parches antes de aplicarlos a los sistemas de cómputo de producción. En algunos casos, instalar un parche en un sistema de la computadora puede tener resultados adversos, como dónde la nueva versión del módulo de código ejecutable identificado contenida en el parche introduce un nuevo error de programación, o donde causa una nueva, interacción impredecible con otro programa que corre en el sistema de cómputo contra el cual es aplicado. En concordancia, a menudo antes de aplicar un parche contra un sistema de producción cuyos datos y funcionamiento son importantes de sostener, el usuario aplica el parche primero contra un sistema de prueba para evaluar si el parche es seguro de aplicar al sistema de producción. Tal comprobación separada de parches agrega carga asociada al "reparchamineto". Adicionalmente, dónde un parche convencional crea un problema--como un problema de compatibilidad de aplicación o una nueva vulnerabilidad – en un momento substancial después de que el parche es aplicado, puede ser difícil de rastrear tales problemas detrás del parche.

[0008] Una desventaja adicional de acercamientos convencionales para remendar se relaciona con el funcionamiento del instalador incluido en el parche. A menudo para reemplazar un módulo de código ejecutable que es parte de ejecutar el programa, el instalador debe terminar la ejecución de

163

ese programa primero. Tampoco, en algunos casos, tal reemplazo puede completarse sin reiniciar el sistema de cómputo. Estos dos pasos pueden causar rupturas sustanciales en el uso del sistema de cómputo remendado.

[0009] Otra desventaja de acercamientos convencionales a remendar involucra intentar remendar un módulo ejecutable para el cual "un parche privado", también llamado " parche caliente ó parche urgente – Hot Fix ", ha sido emitido antes para un subconjunto apropiado de clientes para ese módulo ejecutable. En tales casos, debido a dificultades encontradas en la distribución de parches convencionales que sustituyen nuevas versiones diferentes del módulo del código ejecutable que dependiendo de cada usuario que depende si el usuario ha aplicado el parche caliente, en cambio es típico distribuir un parche convencional simple que suple una sola nueva versión del módulo ejecutable independiente de si el usuario ha aplicado el parche caliente. Si esa nueva versión incluye el parche caliente, el parche impone el parche caliente en clientes no pensados para recibirlo. Si, por otro lado, esa nueva versión no incluye el parche caliente, priva a los clientes pensados para recibir el parche caliente del parche caliente.

[0010] Otra desventaja de acercamientos convencionales a remendar involucra el hecho de que los instaladores para productos de software que confían en un módulo ejecutable particular, como una biblioteca de enlace dinámica particular, a menudo "esconden" el módulo ejecutable guardándolo en una ubicación no-normal en el sistema de archivos (filesystem) del sistema de la computadora objetivo. En concordancia, a veces es difícil o imposible determinar si un sistema designado particular contiene una copia del módulo ejecutable a ser remendado, y, en ese caso, dónde reside en el sistema de archivos (filesystem) del sistema de cómputo designado. También, algunos productos de software mantienen un catálogo de versiones de módulo ejecutables que se han instalado por sus instaladores. Un producto de software puede contar con la exactitud de una indicación en el catálogo de versión de un módulo ejecutable particular. Tal confianza es derrotada donde un parche convencional reemplaza la versión

del módulo ejecutable identificada en el catálogo con una nueva versión del módulo ejecutable sin actualizar el catálogo.

[0011] Otra desventaja de acercamientos convencionales para remiendos sueltos desde el hecho que son imposibles de aplicar en un momento antes de que el módulo ejecutable a ser remendado se instala en el sistema de la computadora designada. Como resultado, si el módulo ejecutable a ser remendado se instala en el sistema de la computadora designada después de que un parche convencional para ese módulo ejecutable se recibe, es improbable que el parche se aplique al módulo ejecutable.

[0012] Otra desventaja de acercamientos convencionales para remendar es que ellos pueden aplicarse típicamente sólo por un usuario autenticado en el sistema de la computadora designada que usa una cuenta administrativa que tiene los permisos de modificación libres o habilitados. Autenticarse en una cuenta administrativa para este propósito puede hacer que el sistema de cómputo designado sea vulnerable a virus presentes en el sistema de la computadora designada que busca modificar aspectos del sistema de la computadora designada y permisos libres requeridos para eso.

[0013] Otra desventaja de acercamientos convencionales a remendar es que esos parches convencionales pueden ser difíciles o imposibles de desactivar, requiriendo pasos tales como reversar el reemplazo de un módulo ejecutable, o reversar una o más modificaciones al registro del sistema.

[0014] En concordancia, un nuevo acercamiento a remendar supera algunas o todas las desventajas de acercamientos convencionales para remendar lo discutido anteriormente tendría utilidad significante.

DESCRIPCIÓN BREVE DE LOS DIBUJOS

[0015] La figura 1 ilustra un ejemplo de un ambiente de sistema de informática conveniente en el cual la facilidad puede implementarse.

[0016] La figura 2 es un diagrama de flujo de datos que muestra un intercambio típico de datos entre los sistemas de la computadora de acuerdo con la facilidad.

[0017] La figura 3 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para recibir y procesar un nuevo parche.

[0018] La figura 4 es un diagrama de estructura de datos que muestra una tabla de parche de muestra típico de aquéllos usados por la facilidad.

[0019] La figura 5 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para poner al día las instrucciones de la configuración para un parche particular.

[0020] La figura 6 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para realizar la aprobación del parámetro especificado por un parche.

DESCRIPCIÓN DETALLADA

[0021] Una facilidad del software para remendar el código de programa de computadora instalado ("la facilidad") es proporcionada. En algunas personalizaciones, la facilidad agrega parámetros de prueba y prueban el resultado manejando las funciones instaladas. En otras personalizaciones, la facilidad agrega varios otros tipos de funcionalidades a las funciones instaladas, en algunos casos en posiciones arbitrarias en los flujos de ejecución de las funciones instaladas.

[0022] En algunas personalizaciones, para cada parche, la facilidad distribuye a cada sistema de computadora a ser remendado—es decir, cada "sistema de computadora" designado – la especificación de un punto en el cual realizar una prueba, la identidad de la prueba a realizar, y cómo actuar en respuesta a uno o más resultados de la prueba diferentes. En algunas personalizaciones, la facilidad proporciona un conjunto formal de validación de parámetro y otras pruebas cuyo uso puede especificarse en los parches. Por ejemplo, un parche puede especificar que, para una función particular,

si un parámetro particular de la función no tiene un cierto valor, la invocación de la función debe fallar antes de que su ejecución substantiva empiece. Otro parche puede especificar que, para una función particular, si un parámetro particular tiene una longitud que excede una longitud máxima especificada, el parámetro debe truncarse a la longitud máxima especificada antes de la ejecución de que la función se permita proceder. Muchos proezas de seguridad se basan en causar que funciones a ser llamadas con valores de parámetro que, mientras ellos no se bloqueen en la versión original del código de una función, causen que la función creer o se aproveche de una condición insegura. En muchos casos, tales ventajas pueden ser prevenidas usando tales parches para prevenir que la función se ejecute con tales valores de parámetro. En algunas personalizaciones, los parches especifican la comprobación de valores de otra manera que los parámetros de la función, tales valores leídos de un archivo o introducidos por el usuario.

[0023] En algunas personalizaciones, un agente remendando automatizado recibe cada parche automáticamente, lo valida, y lo almacena en una tabla de parches para la posible aplicación. En algunas personalizaciones, cada parche se aplica a cualquier instancia del módulo ejecutable a ser remendado que ya está cargado en el sistema de la computadora designada cuando el parche es recibido. Este acercamiento es referido aquí dentro como el "parche caliente – (Hot Fix) ", y permite a los parches ponerse eficazmente inmediatamente al recibirse, y no exige que la facilidad determine donde el módulo ejecutable a ser remendado se guarde en el disco. En algunas personalizaciones, cada parche recibido se aplica a la imagen del disco del módulo ejecutable a ser remendado, para que, cuando la imagen del disco sea cargada un tiempos futuros, la imagen del disco cargada incluya el parche. Este acercamiento referido aquí dentro como el "parche frío – Cold Fix", y permite a los parches ser persistente a través de sesiones múltiples. En algunas personalizaciones, la facilidad realiza el remendando caliente y frío. En algunas personalizaciones, cada parche se

aplica al módulo ejecutable a ser remendado cada vez que el módulo ejecutable a ser remendado es cargado por el cargador del sistema operativo. Este acercamiento es referido aquí dentro como "carga a tiempo del parche – load-time patching." En algunas personalizaciones, cada parche se aplica al módulo ejecutable a ser remendado cada vez que la función a ser remendada es invocada. Este acercamiento es referido aquí dentro como "llamada de interceptación de parche – call-interception patching." Ambas (1), carga a tiempo de parche y llamada a tiempo de parche no exige a la facilidad poder determinar donde el módulo ejecutable a ser remendado se guarda en el disco, (2) la facilidad de reversibilidad lista de un parche particular, y (3) no requiere modificación de la imagen del disco del módulo ejecutable.

[0024] En algunas personalizaciones, la facilidad permite a un usuario o a administrador configurar el funcionamiento de parches que han sido aplicados. Como ejemplos, tal configuración puede incluir, para un parche aplicado en particular: si la prueba especificada por el parche es realizada cuando la ejecución alcanza el punto especificado para el parche; si el resultado de la prueba del manejo especificado por el parche es realizado o ignorado; y/o si el rendimiento de la prueba y/o su resultado es autenticado, desplegado en un mensaje de advertencia, etc. En estas personalizaciones la facilidad permite probar los parches en los sistemas de computadora de producción habilitando y deshabilitando el manejo del resultado autenticado inicialmente. En estas personalizaciones, la facilidad más allá permite la operación del parche a ser autenticado en "un modo verboso" después de habilitar el manejo de su resultado para ayudar con identificar casos en los culaes el parche está creando un problema, como un problema de compatibilidad de aplicación u otro problema IT. Estas personalizaciones también permiten desactivar un parche rápidamente después de ser aplicado si se descubre que el parche está creando problemas. Algunas personalizaciones de la facilidad también permiten que el parche sea

desactivado rápidamente eliminando simplemente el parche del conjunto de parches recibido y guardado en el sistema de computadora designado.

[0025] En algunas personalizaciones, la facilidad usa un "manejo de datos – data-driven" acercamiento de parcheo, en el cual los parches no contienen ningún código, sino datos, como un texto pequeño humano-legible o documento de XML que especifican un punto al cual realizar una prueba, la identidad de la prueba a realizar, y cómo actuar en respuesta a uno o más resultados de prueba diferentes. En tales personalizaciones, un agente de parcheo recibe el manejo de datos (data-driven), y agrega las pruebas y el manejo de la prueba especificados por los parches. En algunas personalizaciones, la facilidad utiliza un acercamiento de parcheo de "manejo de código (code-driven)", y en el cual cada parche contiene un programa corto a ser agregado al módulo ejecutable a ser remendado que realiza la prueba llamando el parámetro normal de la facilidad que prueba las funciones, y que realiza el manejo de la prueba. Usando el parcheo de manejo de datos (data-driven) o el de manejo de código (code-driven), es posible algunas veces direccional todos los sabores de un módulo ejecutable a ser parchado con un solo parche.

[0026] En algunas personalizaciones, la facilidad autentica cada parche para demostrar ambos que (1) el parche es de una fuente aceptado, y (2) no se han modificado los volúmenes del parche desde el tiempo en que el parche se creó por la fuente aceptada.

[0027] En algunas personalizaciones, la facilidad distribuye cada parche para cada sistema de la computadora designada, y un agente de parches en el sistema de la computadora designada determina automáticamente cuales parches para ser aplicados en el sistema de la computadora designada, y cómo serán aplicados, basado en las características del sistema de la computadora designada. Esto releva a los usuarios y administradores de muchas de las cargas convencionalmente asociadas con seleccionar y aplicar los parches, y la carga de mantener una base de datos del inventario exacto y actual. Por ejemplo, estas características

164

pueden incluir qué versión del módulo ejecutable a ser remendado se instala en el sistema de la computadora designada. En estas personalizaciones, la facilidad puede superar el tipo de problemas causado por los parches calientes, distribuyendo parches que especifican el manejo diferente para los parches caliente y no calientes (hot-fixed y un-hot-fixed) de los diferentes sabores del módulo ejecutable en particular, obviando típicamente alguna necesidad de cualquier sacrificio de un parche caliente para un módulo ejecutable particular o hacer que el parche caliente sea ubicuo al remendar ese módulo ejecutable.

[0028] En algunas personalizaciones, el agente de parches almacena cada parche recibido en el sistema designado, independiente de si el módulo ejecutable a ser remendado por un parche particular se instala en el sistema designado cuando ese parche se recibe. Porque la facilidad en muchos casos aplica los parches en respuesta a la carga de un módulo ejecutable a ser remendado o la invocación de una función a ser remendada, la facilidad puede aplicar un parche a un módulo ejecutable que se instaló en el sistema designado después de que ese parche se recibió en el sistema designado. También, los parches pueden sobrevivir la desinstalación y la reinstalación subsiguiente del módulo ejecutable a ser remendado.

[0029] En algunas personalizaciones, el agente de parches es implementado en un servicio del sistema operativo. En estas personalizaciones, la facilidad otorga al agente de parches cualquier permiso exigido para aplicar un parche. Estas personalizaciones reducen el riesgo de seguridad impuesto típicamente cuando los parches convencionales son aplicados, cuando ellos obvian alguna necesidad para un usuario de autenticarse en el sistema de la computadora designada que usa una cuenta administrativa que tiene los permisos de modificación habilitados, y por eso dando a cualquier virus presente en el sistema de la computadora designada una oportunidad mayor de modificar aspectos sensibles del sistema de la computadora designada.

170

[0030] Los parches usados por la facilidad son típicamente relativamente pequeños, y por consiguiente imponen requisitos de recursos modestos para la transmisión y almacenamiento. También, porque los parches usados por la facilidad tienden a modificar el comportamiento del software remendado en un número pequeño de maneras bien-definidas, la facilidad ayuda a reducir el problema de código basura.

[0031] La figura 1 ilustra un ejemplo de un sistema de cómputo conveniente ambiente 100 en el cual la facilidad puede llevarse a cabo. El sistema de cómputo ambiente 100 es sólo un ejemplo de un ambiente de cómputo conveniente y no se piensa en cualquier limitación acerca del alcance de uso o funcionalidad de la facilidad. Tampoco debería ser interpretado el ambiente de cómputo 100 como si tuviera cualquier dependencia o requisito relacionado a cualquier combinación de componentes ilustrados en el ambiente operativo 100 ejemplar.

[0032] La facilidad es operacional con numerosos otros propósitos de propósito general o especial de los ambientes de cómputo del sistema o configuraciones. Los ejemplos de sistemas de cómputo bien conocidos, ambientes, y/o configuraciones que pueden ser convenientes para el uso con la facilidad incluyen, pero no se limita a: las computadoras personales, los servidores, los dispositivos portátiles o portables, los dispositivos de tabla (tablet devices), los sistemas multiprocesador, los sistemas basados en microprocesador, las cajas de juego, electrónicosa de consumo programables, red PCs, los miniordenadores, los sistemas informáticos grandes, ambientes de cómputo distribuidos que incluyen cualquiera de los sistemas anteriores o dispositivos, y similares.

[0033] La facilidad puede describirse en el contexto general de instrucciones de computadora-ejecutables, como los módulos de programa ejecutándose por una computadora. Generalmente, los módulos de programa incluyen las rutinas, programas, objetos, componentes, estructuras de datos, y similares, qué realizan tareas particulares o implementan tipos de datos

abstractos particulares. La facilidad también puede practicarse en ambientes de la informática distribuídos dónde las tareas son realizadas por dispositivos del proceso remotos que se unen a través de una red de comunicaciones. En un ambiente de la informática distribuído, pueden localizarse los módulos del programa en los medios de comunicación de almacenamiento de computadora locales y/o remotos incluso los dispositivos de almacenamiento de memoria.

[0034] Con referencia a la Figurar 1, un sistema ejemplar para implementar la facilidad incluye un dispositivo de cómputo de propósito general en la forma de una computadora 110. Los componentes de la computadora 110 pueden incluir, pero no se limita a, una unidad de proceso 120, un sistema de memoria 130, y un sistema de bus 121 que se acopla a varios componentes del sistema incluso la memoria del sistema a la unidad de proceso 120. El sistema de bujs 121 puede ser cualquiera de varios tipos de estructuras de bus incluso un bus de memoria o controlador de memoria, un bus periférico, y un bus local que usan cualquiera de una variedad de arquitecturas de bus. Por vía del ejemplo, y sin limitación, tales arquitecturas incluyen el bus de la Industria de Arquitectura Estandar (ISA), el bus de Arquitectura de Micro Canal (MCA), el bus ISA extendido (EISA), el bus local de la Asociación de Estándares de Video Electrónico (VESA) el bus local, y el Componente de Interconección Periférico (PCI) también conocido como el bus del entresuelo.

[0035] La computadora 110 incluye una variedad de medios de comunicación típicamente legibles por computadora. Los medios de comunicación legibles por computadora pueden ser cualquier medios de comunicación disponible que puede accederse por la computadora 110 y puede ser incluidos ambos medios de comunicación volátil y no volátil, y los medios de comunicación removibles y no removibles. Por vía del ejemplo, y sin limitación, los medios de comunicación legibles por computadora pueden comprender medios de comunicación de almacenamiento de computadora y medios de comunicación. Los medios de comunicación de

172

almacenamiento de computadora incluyen volátil y no volátil, los medios de comunicación removibles y no removibles implementados en cualquier método o tecnología para el almacenamiento de información como las instrucciones legibles por computadora, las estructuras de datos, los módulos del programa u otros datos. Los medios de comunicación de almacenamiento de computadora incluyen, pero no se limita a, RAM, ROM, EEPROM, memoria de destello (flash memory) u otra tecnología de memoria, CD-ROM, los discos versátiles digitales (DVD) u otro almacenamiento de disco óptico, casetes magnéticos, cinta magnetofónica, almacenamiento de disco magnético u otros dispositivos de almacenamiento magnéticos, o cualquier otro medio que pueda usarse para guardar la información deseada y qué pueda ser accedida por la computadora 110. Los medios de comunicación incluyen las instrucciones legibles típicamente por computadora, las estructuras de datos, módulos de programa u otros datos en señales de datos moduladas como una onda portadora u otro mecanismo de transporte e incluyen cualquier medio de comunicación de entrega de información. El término "señal de datos modulada" significa que tiene uno o más de su conjunto de características o cambios de tal manera que codifica la información de la señal. Por vía del ejemplo, y sin limitación, los medios de comunicación incluyen los medios de comunicación alambrados como una red alambrada o conexión alambrada directa, y los medios de comunicación inalámbricos como el acústico, RF, infrarrojo y otros medios de comunicación inalámbricos. Las combinaciones de cualquiera de los anteriores también debe ser incluido dentro del alcance de medios de comunicación legibles por computadora.

[0036] El sistema de memoria 130 incluye los medios de comunicación de almacenamiento de computadora en la forma de memoria volátil y no volátil y/o tal como la memoria de solo lectura (ROM) 131 y la memoria de acceso aleatorio (RAM) 132. Un sistema básico de entrada/salida 133 (BIOS), conteniendo las rutinas básicas que ayudan a transferir la información entre los elementos dentro de la computadora 110, tal como durante el inicio, que

173

se guarda típicamente en la ROM 131. La RAM 132 típicamente contiene los datos y/o módulos del programa que son inmediatamente accesibles y/o presentemente siendo operados por la unidad de proceso 120. Por vía de ejemplo, y sin limitación, la figure 1 ilustra el sistema operativo 134, la aplicación del programa 135, otros módulos de programa 136 y programas de datos 137.

[0037] La computadora 110 también puede incluir otros medios de comunicación de computadora removibles y no removibles, volátiles y no volátiles. Por vía del ejemplo únicamente, la figure 1 ilustra un controlador de disco duro 141 que lee desde o escribe a un no removible, no volátil de los medios de comunicación magnéticos, una unidad de disco 151 magnética lee desde o escribe a un removible, no volátil disco 152 magnético, y una unidad de disco 155 óptica lee desde o escribe a un trasladable, no volátil disco 156 óptico como un CD ROM u otros medios de comunicación ópticos. Otro removible/no removible, volátil/no volátil medio de comunicación de almacenamiento de computadora que pueden usarse en el ambiente ejemplar que se opera incluyen, pero no se limita a, los casetes de cinta magnetofónica, tarjetas de memoria de destello, discos versátiles digitales, cinta de video digital, RAM de estado sólida, ROM de estado sólido, y similares. La unidad de disco duro 141 se conecta típicamente al sistema de bus 121 a través de una memoria non-removible como la interfaz 140, y la unidad de disco magnética 151 y la unidad de disco óptico 155 son típicamente conectados al sistema de bus 121 por una interfase de memoria removible 150.

[0038] Los controladores y sus medios de comunicación de almacenamiento de computadora asociados, discutidos anteriormente e ilustrados en la figura 1, proporcionan almacenamiento de instrucciones legibles por computadora, las estructuras de datos, los módulos de programa y otros datos para la computadora 110. En la figura 1, por ejemplo, el controlador del disco duro 141 es ilustrado como sistema operativo de almacenamiento 144, los programas de aplicación 145, otros módulos de programa 146 y

programas de datos 147. Note igual que estos componentes pueden ser diferentes o los mismos del sistema operativo 134, el programa de aplicación 135, otros módulos de programa 136, y datos de programa 137. El sistema operativo 144, el programa de aplicación 145, otros módulos de programa 146, y datos de programa 147 a los cuales son dados aquí dentro números diferentes para ilustrar que, en un mínimo, ellos son copias diferentes. Un usuario puede entrar las órdenes e información en la computadora 110 a través de los dispositivos de entrada como tabla, o digitalizador electrónico, 164, un micrófono 163, un teclado 162 y un dispositivo de apuntamiento 161, normalmente llamado ratón, trackball o almohadilla de tacto. Otros dispositivos de entrada no mostrados en la figura 1 pueden incluir una palanca de mando, la almohadilla de juego, plato satélite, escáner, o similares. Se conectan a menudo éstos y otros dispositivos de entrada a la unidad de proceso 120 a través de una interfaz de entrada de usuario 160 que se acopla al bus del sistema, pero puede conectarse por otra interfaz y estructuras de bus, como un puerto paralelo; un puerto de juego o un bus de serie universal (USB). Un monitor 191 u otro tipo de dispositivo de despliegue también se conecta al sistema de bus 121 vía una interfaz, como una interfaz de video 190. El monitor 191 también puede integrarse con una pantalla de tacto o similares. Note que el monitor y/o el tablero de pantalla de tacto puede acoplarse físicamente a un alojamiento en el cual el dispositivo de cómputo 110 está incorporado, como en un computador personal tipo tabla. Además, las computadoras como el dispositivo de cómputo 110 también puede incluir otros dispositivos de rendimiento periféricos como parlantes 195 e impresora 196 que pueden conectarse a través de una interfaz periférica de salida 194 o similares.

[0039] La computadora 110 puede operar en un ambiente conectado a una red de computadoras que usa conexiones lógicas o a las computadoras más remotas, como una computadora remota 180. La computadora 180 remota puede ser una computadora personal, un servidor, un enrutador, una red de PCs, un dispositivo par u otro nodo de la red común, y

175

típicamente incluye muchos o todos los elementos descritos arriba relacionados a la computadora 110, aunque sólo un dispositivo de almacenamiento de memoria 181 se ha ilustrado en la figura 1. Las conexiones lógicas pintadas en la figura 1 incluyen una red de área local (LAN) 171 y una red de área amplia (WAN) 173, pero también puede incluir otras redes. Tales ambientes de gestión de redes son comúnes en las oficinas, en las redes de computadores de empresas grandes, intranets e Internet. Por ejemplo, en la facilidad presente, el sistema de cómputo 110 puede comprender la fuente de máquina de la cual los datos están siendo migrados, y la computadora remota 180 puede comprender la máquina del destino. Note sin embargo que esas máquinas fuente y destino no necesita ser conectados por una red o cualquier otro medio, pero en cambio, pueden emigrarse los datos por cualquier medio de comunicación capaz de ser escrito por la plataforma de fuente y pueden leerse por la plataforma o plataformas destino.

[0040] Cuando es usado en un ambiente de red LAN que conecta una red de computadoras, la computadora 110 se conecta a la LAN 171 a través de una interfaz de red o adaptador 170. Cuando es usado en un ambiente de redes WAN, la computadora 110 incluye un módem 172 u otro medio típicamente para establecer las comunicaciones sobre la WAN 173, como Internet. El módem 172 puede ser interno o externo y puede conectarse al sistema de bus 121 vía la interfaz de usuario 160 u otro mecanismo apropiado. En un ambiente de red conectado, los módulos de programa graficados relacionados a la computadora 110, o parte de ello, puede guardarse en el dispositivo de almacenamiento de memoria remoto. Por vía de ejemplo, y sin limitación, la figure 1 ilustra el programa de aplicación remota 185 como residente en el dispositivo de memoria 181. Se apreciará que las conexiones de red mostradas son medios ejemplares y otros medios para establecer un eslabón de comunicaciones entre las computadoras pueden usarse.

177

[0041] Mientras varias funcionalidades y datos son mostradas en la figura 1 como residentes en sistemas de cómputo particulares que son colocados de una manera particular, estos experimentados en el arte apreciarán que tales funcionalidades y datos pueden distribuirse de varias otras maneras por los sistemas de cómputo en los diferentes arreglos. Mientras los sistemas de cómputo configurados como los descritos anteriormente son típicamente usados para apoyar el funcionamiento de la facilidad, una habilidad ordinaria en el arte apreciará que la facilidad puede ser implementada usando dispositivos de varios tipos y configuraciones, y teniendo varios componentes.

[0042] La figura 2 es un diagrama de flujo de datos que muestra un intercambio típico de datos entre los sistemas de cómputo de acuerdo con la facilidad. Los sistemas de cómputo mostrados en la figura 2 (sistemas de cómputo 210, 220, 221, 222, 230, 231, y 232) típicamente tienen algunos o todos los componentes mostrados y discutidos junto con la figura 1. En el servidor de distribución de parches, la facilidad genera uno o más parches. Estos parches 201 se envían del servidor de distribución de parche a uno o más servidores de administración, como los servidores de administración 220 y 230. Cada servidor de administración, a su vez, adelanta los parches a uno o más sistemas de cómputo designados, como la computadora designada sistemas 221, 222, 231, y 232. En algunas personalizaciones (no mostradas), el servidor de distribución de parches envía los parches directamente a uno o más sistemas de cómputo designados, vía una ruta más indirecta que a través de un solo servidor de administración. Se procesan los parches recibidos en un sistema de cómputo designado como descrito en mayor detalle debajo. Un servidor de administración también puede enviar comandos de la configuración del parche 202 a uno o más sistemas de cómputo designados que aplican los comandos de configuración del parche que reconfiguran la operación de parches particulares. Como se discute en mayor detalle debajo, un parche puede desactivarse completamente; si un parche no es invalido, la notificación de

177

su funcionamiento y su manejo de resultado de prueba puede habilitarse o deshabilitarse a cada uno independientemente. Cuando la notificación de su funcionamiento se habilita, pueden desplegarse las notificaciones o pueden guardarse localmente en el sistema de la computadora designada, o pueden enviarse como notificaciones 203 a un servidor de administración apropiado.

[0043] La figura 3 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para recibir y procesar un nuevo parche. En el paso 301, la facilidad recibe un parche. El parche recibido en el paso 301 puede ser un parche de manejo de datos o un parche de manejo de código. Una muestra de parche de código de datos se muestra en la tabla 1 debajo.

```
1    <Softpatch Patch="Q382429">
2     <AffectedApplication AffectedExe="sqlservr.exe">
3      <AffectedVersion Version="9.*">
4       <AffectedModules Name="SQLSORT.DLL">
5        <Version "8.0.*, 9.*">
6              <Function Name="SsrpEnumCore" Address="0x0802E76B"
7                  Param="2" Paramtype="LPSTR">
8               <Filter MaxByteLength="60" />
9               <Resolution ActionType="BOOL" Action="FALSE" />
10             </Function>
11       </Version>
12       <Version "10.*, 11.*">
13             <Function Name="SsrpEnumCore" Address="0x0802D283"
14                 Param="2" Paramtype="LPSTR">
15              <Filter MaxByteLength="128" />
16              <Resolution ActionType="BOOL" Action="FALSE" />
17             </Function>
18       </Version>
19      </AffectedModules>
20     </AffectedVersion>
21    </AffectedApplication>
22
23    <Signature Hash="MD5" Signature="C509-64AA-9161-8C52-
24     9F6D-BF5A-AEF2-ECE1-0038-34D1"/>
25   </Softpatch>
```

TABLA 1

[0044] La línea 1 contiene un único identificador para el parche. La línea 2 identifica la aplicación afectada por el parche. La línea 3 identifica la versión de la aplicación afectada por el parche. La línea 4 identifica el módulo ejecutable afectado por el parche. La línea 5 identifica dos versiones del módulo ejecutable afectado–versiones 8.0. * y 9. *– para los cuales las direcciones del parche son proporcionados. Las líneas 6-10 contienen las direcciones de parche para estas dos versiones del módulo ejecutable. Las líneas 6-7 identifican la función a ser remendada, su dirección en el módulo ejecutable, su parámetro a ser probado por el parche, y el tipo del parámetro a ser probado. La línea 8 indica que el parámetro identificado en las líneas 6-7 debe probarse para determinar si su longitud excede 60 bytes. La línea 9 indica que la invocación de la función debe fallar si la prueba tiene éxito. La línea 12 identifica dos ó más versiones del módulo ejecutable afectado – las versiones 10. * y 11. *– para las cuales las direcciones del parche se proporcionan. Las líneas 13-17 contienen las direcciones del parche para estas dos versiones del módulo ejecutable. Las líneas 13-14 identifican la función a ser remendada, su dirección en el módulo ejecutable, su parámetro a ser probado por el parche, y el tipo del parámetro a ser probado. Puede verse que la dirección de la función a ser remendada en el módulo ejecutable identificado en las líneas 13-14 para versiones 10. * y 11. * es diferente de la dirección de la función a ser remendada identificada en las líneas 6-7 para las versiones 8.0. * y 9. *. La línea 15 indica que el parámetro identificado en las líneas 13-14 que debe probarse para determinar si su longitud excede 128 bytes. La línea 16 indica que la invocación de la función debe fallar si la prueba tiene éxito. El parche puede especificar una variedad de tipos de acción de manejo de resultado, incluso el fracaso de la invocación de la función remendada, levantando una excepción, terminando el proceso en que el módulo ejecutable remendado está ejecutándose, o corrigiendo el valor

erróneo (tal como truncando una cadena demasiado larga). Las líneas 23-25 contienen una firma para el parche que identifica la fuente del parche y verifica que el parche no ha sido alterado desde de su fuente.

[0045] La Tabla 2 debajo contiene una versión de manejo de código (code-driven) del parche mostrado anteriormente en la Tabla 1.

```
1     00411A7E  push    3Ch
2     00411A80  mov     eax,dword ptr [str]
3     00411A83  push    eax
4     00411A84  call    ValidateStringLength (411082h)
5     00411A89  add     esp,8
6     00411A8C  movzx   ecx,al
7     00411A8F  test    ecx,ecx
8     00411A91  je      411A9Ah
9     00411A93  jmp     foo+2 (411AD2h)
10    00411A9A  xor     eax,eax
11    00411A9C  ret
```

TABLA 2

[0046] Las líneas 1-3 empujan los parámetros para la función de prueba hacia la pila. La línea 4 llama la función de prueba. Las líneas 5-8 dependen del código de retorno para la función de comprobación. Si la función de prueba tuviera éxito, la línea 9 retorna para empezar ejecutando el cuerpo de la función remendada. Si la función de prueba fallara, las líneas 10-11 empujan un código de resultado de fracaso hacia la pila y retorna desde la función de reparcheo al color de la función remendada. Para la legibilidad, la Tabla 2 omite ciertos detalles presentes en algunos parches de manejo de código (code-driven), incluso una firma comprobable, las instrucciones que prueban para los valores actuales de las banderas de configuración del parche, y las instrucciones relocalizadas desde el principio del código para la función de reparcheo.

[0047] En algunas personalizaciones, los parches de ambos tipos pueden contener la información adicional, incluyendo para cada uno de una o más versiones del módulo ejecutable para ser parchado, una firma de archivo que puede usarse para validar que un caso particular del módulo ejecutable es una copia apropiada de esa versión. Tal archivo de firma puede ser por

ejemplo, un tamaño o una verificación para la versión del módulo ejecutable completo, o el código esperado a ocurrir en un punto particular en el módulo ejecutable, como el desplazamiento al que el módulo ejecutable será remendado.

[0048] En el paso 302, si el parche se firma con una firma válida, entonces la facilidad continúa al paso 303, el resto de la facilidad continúa al paso 301 para recibir el próximo parche. En el paso 303, la facilidad agrega el parche a una tabla del parche local. En el paso 304, la facilidad inicializa la configuración inicial del parche, como enviándolo a una configuración predefinida.

[0049] La figura 4 es un diagrama de estructura de datos que muestra una tabla de parche de muestra típico de aquéllos usados por la facilidad. La tabla del parche 400 contiene las filas, como las filas 401 y 402, cada una dividida en las columnas siguientes: una columna identificadora del parche 411, conteniendo el identificador del parche extraído del parche,; una columna del módulo ejecutable 412, conteniendo la información que identifica el módulo ejecutable a ser reparchado, tal como su nombre,; una columna de versiones de módulo ejecutables 413 que identifica todas las versiones del módulo ejecutable identificado en la columna 412 al que el parche aplica; una columna habilitada de rendimiento de prueba 414, conteniendo el valor de la configuración actual cual por si la prueba especificada por el parche debería realizarse cada vez que la función de reparchado es llamada; una columna habilitada de notificación de rendimiento de prueba 415, conteniendo el valor de la configuración actual por si una notificación debe generarse cada vez que la prueba del parche se ha realizado; una columna habilitada de notificación de resultado del prueba 416, conteniendo el valor de la configuración actual por si una notificación debería generarse cada vez que la prueba del parche tiene éxito; una columna habilitada de manejo de resultado de la prueba 417, conteniendo el valor de la configuración actual por si el manejo de resultado del parche debe llevarse a cabo cuando fallan las pruebas del parche y una

columna del parche 418 que contiene un apuntador al propio parche, especificando una prueba y un manejo de resultado de la prueba para ser realizado cada vez que falla la prueba. En algunas personalizaciones, en lugar de contener un indicador como el parche mostrado, la columna del parche 418 contiene cada parche directamente. Una tabla de parche particular puede contener o puede apuntar a parches de varios tipos, como todos los parches de manejo de código, todos los parches de manejo de datos, o una combinación de parches de manejo de código y de manejo de datos.

[0050] En el paso 305, una vez la facilidad ha agregado el parche recibido a la tabla de parches y ha inicializado sus configuraciones, el parche está disponible para ser aplicado automáticamente por la facilidad en el módulo ejecutable. En el paso 305, la facilidad puede utilizar una variedad de acercamientos para aplicar el parche, incluyendo aquéllos descritos en la aplicación incorporada por la referencia, así como la función de tiempo real llamada interceptación, y/o la reescritura de código de (1) módulos ejecutables ya listos cargados, (2) uno o más imágenes del módulo ejecutable del disco, o (3) instancias de módulos ejecutables cargados por el cargador del sistema operativo. Después del paso 305, la facilidad continúa en el paso 301 para recibir el próximo parche.

[0051] La figura 5 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para poner al día las instrucciones de la configuración para un parche en particular. En el paso 501, la facilidad recibe las instrucciones de la configuración para un parche en particular, tal como de un administrador. En algunas personalizaciones, tales instrucciones de configuración pueden ser generadas por administradores que usan las políticas de grupo. En el paso 502, la facilidad actualiza que el parche es la configuración de la Tabla de parches de acuerdo con las instrucciones recibidas. Después del paso 502, la facilidad continua en el paso 501 para recibir las próximas instrucciones de configuración.

[0052] La figura 6 es un diagrama de flujo que muestra los pasos típicamente realizados por la facilidad para realizar la aprobación del parámetro especificado por un parche. En el paso 601, una función de parcheo es llamada. En el paso 602, si la prueba es habilitada para el parche que afecta la función llamada, entonces la facilidad continúa en el paso 603, sino, la facilidad continúa en el paso 601 para procesar la próxima llamada a una función de parcheo. En el paso 603, si la notificación de rendimiento de prueba se habilita para el parche, entonces la facilidad continúa en el paso 604, sino la facilidad continúa en el paso 605. En el paso 604, la facilidad genera una notificación de que la prueba fue realizada. En los pasos 604, 608, y 610 pueden involucrar el despliegue o almacenamiento de una indicación en el sistema de la computadora designada de que la prueba fue satisfecha, y/o transmitiendo tal indicación a un sistema de cómputo remoto para el despliegue o registro allí.

[0053] En el paso 605, la facilidad realiza la prueba de aprobación especificada por el parche. En algunas personalizaciones, el paso 605 involucra la llamada de un grupo de rutinas normales utilizada spor la facilidad para probar. En el paso 606, si la prueba realizada en el paso 605 fue satisfecha, entonces la facilidad continúa en el paso 601, sino la facilidad continúa en el paso 607. En el paso 607, si la notificación de resultado de prueba es habilitado para el parche, entonces la facilidad continúa en el paso 608, sino la facilidad continúa en el paso 609. En el paso 608, la facilidad genera una notificación de que la prueba no fue satisfecha. En el paso 609, si el manejo del resultado de la prueba se habilita para el parche, entonces la facilidad continúa en el paso 610, sino, la facilidad continúa en paso 601. En paso 610, la facilidad realiza el prueba resultado manejando especificó por el parche. Después del paso 610, la facilidad continúa en el paso 601.

[0054] Será apreciado por estos experimentado en el arte que la facilidad arriba descrita puede adaptarse correctamente o puede extenderse de varias maneras. Por ejemplo, la facilidad puede usarse para aplicar una

variedad de tipos diferentes de parches en una variedad de maneras a una variedad de situaciones en los módulos ejecutables de una variedad de tipos para una variedad de propósitos. También, mientras son descritos los parches aquí dentro de cómo contener pruebas de aprobación de valor que indican un problema cuando ellos fallan, la facilidad también puede ser implementada usando la pruebas de invalidez de valor que indican un problema que puede llevarse a cabo cuando ellos tienen éxito. En algunas personalizaciones, cada prueba se acompaña por una indicación de si el éxito o el fracaso indica un problema. Mientras la descripción anterior hace referencia a las personalizaciones preferidas, el alcance de la invención se define solamente por las demandas que siguen y los elementos citados aquí dentro.

LAS DEMANDAS

Nosotros Demandamos:

[c1] 1. Un método en un sistema de cómputo para aumentar el software, comprendiendo:

Recibir en un sistema de cómputo designado una especificación de aumento que especifica (a) una función a ser aumentada, la función especificada estando entre el software ejecutado en el sistema de cómputo, (b) un parámetro de la función a ser aprobado, (c) una prueba para aplicar al parámetro especificado, y (d) una modificación para realizar al comportamiento de función si la prueba especificada no es satisfecha por el parámetro especificado; y

Cuando la función especificada es invocada en el sistema de la computadora designada, si la prueba especificada no es satisfecha por el parámetro especificado, realizar la modificación especificada a la función especificada.

HBA

[c2] 2. El método de la demanda 1 en donde la modificación especificada por la especificación de aumento está previniendo la ejecución de la función especificada.

[c3] 3. El método de la demanda 1 de la modificación especificada por la especificación de aumento está ejecutando la función especificada después de la alteración del parámetro especificado.

[c4] 4. El método de la demanda 1, en donde una pluralidad de especificaciones de aumento se recibe en el sistema de la computadora designada.

[c5] 5. El método de la demanda 4, más allá comprende el manteniendo de todas las especificaciones de aumento recibidas en una estructura de datos.

[c6] 6. El método de la demanda 1 en qué ninguna intervención de usuario subsiguiente se requiere para recibir la especificación de aumento para realizar la modificación especificada a la conducta de la función especificada.

[c7] 7. El método de la demanda 1 en donde la especificación de aumento especifica una prueba para aplicar al parámetro especificado identificando un parámetro que prueba la función para invocar de quien el código no es incluido en la especificación de aumento.

[c8] 8. El método de la demanda 1 en donde la especificación de aumento especifica una prueba a ser aplicada al parámetro especificado identificando un parámetro que prueba la función para invocar que fue instalado antes de la especificación de aumento recibida.

[c9] 9. El método de la demanda 1 en donde la especificación de aumento contiene el código.

[c10] 10. El método de la demanda 9 en donde el código contenido por la especificación de aumento incluye código que invoca una función de prueba de parámetro cuyo código no es incluido en la especificación de aumento.

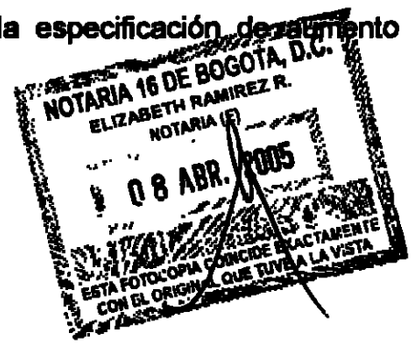
[c11] 11. El método de la demanda 1 en donde la especificación de aumento contiene texto que identifica una función de prueba de parámetro cuyo código no está incluido en la especificación de aumento.

[c12] 12. El método de la demanda 11 en donde una función de prueba de parámetro identificada por el texto contenido por la especificación de aumento está contenida por la especificación de aumento.

[c13] 13. El método de la demanda 11 en donde una función de prueba de parámetro identificada por el texto no contenido por la especificación de aumento no está contenida por la especificación de aumento.

[c14] 14. El método de la demanda 1 más allá comprende proporcionar una interfaz para configurar un estado para controlar la operación de la especificación de aumento,
y en donde la modificación especificada para el comportamiento de la función especificada es realizada solo cuando el estado es un estado habilitado.

[c15] 15. El método de la demanda 14 en donde la interfaz proporcionada permite que la operación de la especificación de aumento sea configurada localmente.



[c16] 16. El método de la demanda 14 en donde en donde la interfaz proporcionada permite la operación de la especificación de aumento sea configurada remotamente.

[c17] 17. El método de la demanda 14 en donde donde la interfaz proporcionada permite la operación de la especificación de aumento sea configurada de acuerdo con una política.

[c18] 18. El método de la demanda 14 en donde la prueba especificada es realizada antes de que cualquier código sustantivo de la función especificada se ejecute.

[c19] 19. El método de la demanda 1 más allá comprende proporcionar una interfaz para la configurando de advertencias asociadas con la especificación de aumento, y donde, si la interfaz proporcionada es usada para habilitar las advertencias, generando una advertencia cada vez que la prueba especificada por la especificación de aumento falla.

[20] 20. El método de la demanda 1, más allá comprende proporcionar una interfaz para configurar las notificaciones asociadas con la especificación de aumento, y en donde, si la interfaz proporcionada se usa para habilitar las notificaciones, generando una notificación cada vez que la prueba especificada por la especificación de aumento es realizada.

[c21] 21. El método de la demanda 20, más allá comprende consolidar las notificaciones generadas en una sola notificación consolidada.

[c22] 22. El método de la demanda 1 en donde la especificación de aumento recibida es firmada,
el método más allá comprende verificar la firma de la especificación de aumento.

y en donde la modificación especificada para el comportamiento de la función especificada sólo es realizado si la verificación de la firma de la especificación de aumento tuvo éxito.

[c23] 23. El método de la demanda 22 en donde la modificación especificada para el comportamiento de la función especificada es realizado solo si el autenticador de la firma de la especificación de aumento coincide con un autenticador del software que contiene la función especificada.

[c24] 24. El método de la demanda 1, más allá comprende realizar la prueba especificada en respuesta a descubrir que la función especificada ha sido invocada, sin alterar el código de la función especificada.

[c25] 25. El método de la demanda 1, más allá comprende alterar el código de la función especificada para realizar la prueba especificada cuando la función especificada es invocada.

[c26] 26. El método de la demanda 25 en donde el código de la función especificada es alterada insertando en el código de la función especificada un salto a una rutina separada que realiza la prueba especificada.

[c27] 27. El método de la demanda 1 en donde la función especificada es proporcionada en un módulo ejecutable distinguido que es posible cargar usando a un cargador, el método más allá comprende registrar con el cargador para tener una rutina que realiza el llamado de la prueba especificada antes de que la función especificada sea ejecutada en respuesta a cada invocación de la función especificada.

[c28] 28. Un sistema de cómputo que habilita el aumento de software presente en el sistema de cómputo, comprendiendo:

un dispositivo de almacenamiento que contiene el software

188

un receptor del parche que recibe en el sistema de cómputo un parche especificando (a) un punto al cual el software será aumentado, (b) un valor asociado con la función a ser probada en el punto especificado, (c) una prueba para aplicar al valor especificado, y (d) una modificación para realizar al comportamiento del software si la prueba especificada no es satisfecha por el valor especificado; y

un agente parches que inyecta el código en el software al punto especificado tal que, cuando la ejecución del software alcanza el punto especificado en el sistema de la computadora designada, si la prueba especificada no es satisfecha por el valor especificado, la modificación especificada al comportamiento del software es realizada.

[c29] 29. Un medio legible por computadora cuyos contenidos causan a un sistema de cómputo designado para realizar un método para adicionar la aprobación de valor del software disponible en el sistema de cómputo designado, comprendiendo:

recibir en el sistema de cómputo una especificación de aumento que especifica (a) una función para la cual la aprobación de valor será agregada, la función especificada que está entre el software disponible y el sistema de cómputo, (b) los datos a ser probados que existen en el sistema de cómputo designado durante la ejecución de la función, (c) una prueba para aplicar a los datos especificados, y (d) una modificación para realizar al comportamiento de la función si la prueba especificada no es satisfecha por los datos especificados; y

cuando la función especificada se invoca en el sistema de la computadora designada, si la prueba especificada no es satisfecha por los datos especificados, realizando la modificación especificada al comportamiento de la función especificada.

[c30] 30. El medio legible por la computadora de demanda 22 en donde la prueba especificada es una prueba que siempre es satisfecha, independientemente del valor de los datos especificados.

189

[c31] 31. Una señal de datos generada llevando un parche de la estructura de datos, comprendiendo:

información que identifica una función a la cual la aprobación de valor será agregada;

información que identifica datos a ser probados que existen durante la ejecución de la función;

información que identifica una prueba para aplicar a los datos especificados; y para

información que identifica una modificación para realizar al comportamiento de la función si la prueba especificada no es satisfecha por los datos especificados,

Así, si la señal de datos es recibida en un sistema de la computadora designada en el cual la función es ejecutada, los contenidos de la estructura de datos puede ser usada para realizar la modificación especificada al comportamiento de la función especificada si la prueba especificada no es satisfecha por los datos especificados cuando la función especificada se invoca en el sistema de la computadora designada.

[c32] 32. La señal de datos generada de la demanda 31 en donde la señal de datos generada se transmite entre los sistemas de cómputo.

[c33] 33. La señal de datos generada de la demanda 31 en donde La señal de datos generada se transmite dentro de un sistema de cómputo.

[c34] 34. Un método para agregar la aprobación de valor al software disponible en el sistema de cómputo designado, comprendiendo:

recibir en el sistema de cómputo una especificación del elemento que especifica (a) software para el cual la aprobación de valor será agregada, (b) un punto en el software especificado en el cual la aprobación de valor será agregada, (c) los datos a ser probados que existen en el sistema de la computadora designada durante la ejecución del software especificado en

el punto especificado, (d) una prueba para aplicar a los datos especificados, y (e) una modificación para realizar al comportamiento del software si la prueba especificada no está satisfecha por los datos especificados;

permitir a un usuario que configure un modo de operación para la especificación de aumento recibida; y

cuando el software especificado se ejecute en el punto especificado en el sistema de la computadora designada, aplicando la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida.

[c35] 35. El método de la demanda 34 en donde el usuario configura un modo operacional activo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende, si la prueba especificada no está satisfecha por los datos especificados, realizando la modificación especificada al comportamiento del software especificado.

[c36] 36. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico activo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende, si la prueba especificada no está satisfecha por los datos especificados, ambos (1) realizar la modificación especificada al comportamiento del software especificado, y (2) generar una notificación de que la prueba especificada no es satisfecha por los datos especificados.

[c37] 37. El método de la demanda 34 en donde el usuario configura un modo operacional activo difuso para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

si la prueba especificada no es satisfecha por los datos especificados, realizando la modificación especificada al comportamiento del software especificado; y

generar una notificación de que la prueba especificada fue realizada.

[c38] 38. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico difuso activo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

si la prueba especificada no es satisfecha por los datos especificados, ambos (1) realizar la modificación especificada a la conducta del software especificado, y (2) generar una notificación de que la prueba especificada no es satisfecha por los datos especificados; y

generar una notificación de que la prueba especificada fue realizada.

[c39] 39. El método de la demanda 34 en donde el usuario configura un modo operacional inactivo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados.

[c40] 40. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico inactivo I para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

192

omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados; y

si la prueba especificada no está satisfecha por los datos especificados, generando una notificación de que la prueba especificada no es satisfecha por los datos especificados.

[c41] 41. El método de la demanda 34 en donde el usuario configura un modo operacional de diagnóstico inactivo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados; y

generar una notificación de que la prueba especificada fue realizada.

[c42] 42. El método de la demanda 34 en donde el usuario configura modo operacional de diagnóstico inactivo para la especificación de aumento, y en donde aplicar la especificación de aumento recibida de una manera consistente con el modo operacional configurado para la especificación de aumento recibida comprende:

omitir para realizar la modificación especificada al comportamiento del software especificado, independiente de si la prueba especificada es satisfecha por los datos especificados;

generar una notificación de que la prueba especificada fue realizada; y si la prueba especificada no es satisfecha por los datos especificados, generando una notificación de que la prueba especificada no es satisfecha por los datos especificados.

ABSTRACTO

Una facilidad para el software de aumento en un sistema de cómputo designado es descrito. La facilidad recibe y la especificación de aumento en el sistema de la computadora designada. La especificación de aumentos especifica: (a) una función a ser aumentada, (b) un parámetro de la función a ser probado, (c) una prueba para aplicar al parámetro especificado, y (d) y la modificación para realizar al comportamiento de la función si la prueba especificada no es satisfecha por el parámetro especificado. Cuando la función especificada es invocada en el sistema de la computadora designada, si lo probado especificado no es satisfecho por el parámetro especificado, la facilidad realiza la modificación especificada al comportamiento de la función especificada.

SEGUNDA REIVINDICACIÓN

Reivindicamos:

[c43] 1. Un método de un sistema de computación que incluye:
recepción de un paquete de parches distintivos para modificar el comportamiento de un programa instalado en un sistema de computación;
la extracción automática de la información de la aplicación de dicho paquete de parches distintivos (1), que identifica una porción distintiva de un programa distintivo al cual se aplica el parche, y (2) información del comportamiento del parche que especifica la manera en la que se modifica el comportamiento de la porción del programa distintivo; y
la adición automática de una entrada distintiva a una tabla de parches, la cual incluye la información de la aplicación del parche de extracción y la información del comportamiento del parche.

[c44] 2. El método de la reivindicación 1, que además incluye la utilización de contenidos de la entrada distintiva para modificar el comportamiento de la porción distintiva del programa distintivo.

[c45] 3. El método de la reivindicación 2, por el que los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva del programa distintivo en un momento en que ningún usuario administrativo se encuentre utilizando el sistema de computación.

[c46] 4. El método de la reivindicación 2, por el que los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva del programa distintivo sin necesidad de autorización alguna de usuario.

[c47] 5. El método de la reivindicación 2, por el que los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva del programa distintivo sin determinar la ubicación donde el programa distintivo se guarda de manera persistente.

[c48] 6. El método de la reivindicación 2, por el que los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva del

programa distintivo como respuesta de la instalación de dicho programa distintivo.

[c49] 7. El método de la reivindicación 2, por el que los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva del programa como respuesta a la ejecución de la porción distintiva del programa distintivo.

[c50] 8. El método de la reivindicación 2, por el que se instalan dos instancias del programa distintivo, y los contenidos de la entrada distintiva se utilizan para modificar el comportamiento de la porción distintiva de las dos instancias del programa distintivo.

[c51] 9. El método de la reivindicación 1, que además incluye:

la determinación si el programa distintivo está instalado en el sistema de computación al momento en que se extrae la información de la aplicación del parche; y

si el programa distintivo está instalado en el sistema de computación, utilizando la información extraída de la aplicación del parche para modificar el comportamiento del programa distintivo instalado de acuerdo con la información del comportamiento del parche.

[c52] 10. El método de la reivindicación 1, que además incluye la eliminación de la entrada distintiva de la tabla del parche para prevenir que el comportamiento de la porción distintiva del programa distintivo se modifique de acuerdo con la información del comportamiento del parche contenida en la entrada distintiva.

[c53] 11. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de un valor.

[c54] 12. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de parámetros.

[c55] 13. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la

196

porción distintiva del programa distintivo para llevar a cabo la validación de un valor leído de un archivo.

[c56] 14. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de un valor ingresado por el usuario.

[c57] 15. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de un valor recibido de uno o más paquetes de la red.

[c58] 16. El método de la reivindicación 1, por el que la información extraída del comportamiento del parche especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de parámetros por medio de la utilización de una función de validación distintiva.

[c59] 17. El método de la reivindicación 16, por el que una entrada en la tabla de parches distinta de la entrada distintiva, también incluye la información del comportamiento del parche que especifica la modificación del comportamiento de la porción distintiva del programa distintivo para llevar a cabo la validación de parámetros utilizando la función de validación distintiva.

[c60] 18. El método de la reivindicación 16, que además incluye el código de ejecución por el cual la función de validación distintiva modifica el comportamiento de la porción distintiva del programa distintivo, siendo que este código de ejecución no está incluido dentro del paquete de parches distintivos.

[c61] 19. Un medio reconocido por el computador, cuyo contenido lleve a que el sistema de computación realice un método que comprende:

la recepción de un paquete de parches distintivos que modifican el comportamiento de una entidad de programación en un sistema de computación;

la extracción automática de la información de la aplicación de dicho paquete de parches distintivos (1), que identifica una entidad de programación distintiva al cual se aplica el parche, y (2) información del comportamiento del parche que especifica la manera en la que se modifica el comportamiento de la porción de la

entidad de programación; y

la adición automática de una entrada distintiva a una tabla de parches, la cual incluye la información de la aplicación del parche de extracción y la información del comportamiento del parche.

[c62] 20. El medio reconocido por el computador de la reivindicación 19 por el cual la entidad de programación distintiva tiene varios comportamientos, de los cuales sólo un subconjunto adecuado está incluido en la información de comportamiento del parche que especifica la modificación, y por el cual el paquete de parches distintivos recibidos no contiene información sobre los comportamientos de la entidad de programación distintiva que la información del comportamiento del parche no especifica la modificación.

[c63] 21. Una o más memorias de computador que guardan de manera colectiva una estructura de de información de tabla de parche, que incluye varias entradas de parche, cada una de las cuales contiene:

información de la aplicación del parche, que identifica una porción distintiva de una entidad de programación distintiva al cual se debe aplicar el parche; e

información del comportamiento del parche, que especifica la manera en que se debe modificar el comportamiento de la porción distintiva de la entidad de programación distintiva, tal que, para una entrada de parche dada, los contenidos de dicha entrada pueden ser utilizadas para modificar el comportamiento de la porción distintiva de la entidad de programación distintiva en la manera especificada.

[c64] 22. Las memorias de computador de la reivindicación 21, por las cuales la información de la aplicación del parche identifica un módulo ejecutable asociado con la entidad de programación.

[c65] 23. Las memorias de computador de la reivindicación 22, por las cuales la información de la aplicación del parche identifica una posición en el módulo ejecutable identificado.

[c66] 24. Las memorias de computador de la reivindicación 23, por las cuales la información de la aplicación del parche identifica los contenidos esperados en la posición identificada en el módulo ejecutable identificado.

[c67] 25. Las memorias de computador de la reivindicación 21, por las cuales una entrada de parche seleccionada contiene varias instancias de información de aplicación del parche, cada una de las cuales identifica una porción distintiva de una versión diferente de la entidad de programación distintiva.

[c68] 26. Las memorias de computador de la reivindicación 25, por las cuales una primera instancia de información de aplicación del parche identifica una versión de la entidad de programación distintiva al cual se aplicó una situación activa, y por el cual una segunda instancia de información de aplicación del parche identifica una versión de la entidad de programación distintiva al cual no se aplicó una situación activa.

[c69] 27. El método de la reivindicación 26, por el cual la entrada del parche seleccionado puede ser aplicado a (1), la versión de la entidad de programación distintiva al cual se aplica una situación activa, (2) la versión de la entidad e programación distintiva al cual no se aplica la situación activa seleccionada, o (3) ambos, sin reemplazar la versión de la entidad de programación distintiva al cual se aplica la situación activa seleccionada con la versión de la entidad de programación distintiva al cual no se aplica la situación activa, y sin reemplazar la versión de la entidad de programación distintiva al cual no se ha aplicado la situación activa con la versión de la entidad de programación distintiva al cual se aplicó la situación activa seleccionada; y

[c70] 28. El método de la reivindicación 24, por el cual la entrada del parche seleccionado puede ser desplegada a un sistema de computación objetivo por un administrador, sin importar qué versión de la entidad de programación distintiva se ejecute en el sistema de computación objetivo.

[c71] 29. El método de la reivindicación 21, por el cual la información del comportamiento del parche de una entrada de parche seleccionada contiene un código utilizable para modificar el comportamiento de la porción distintiva de la entidad de programación distintiva.

[c72] 30. El método de la reivindicación 21, por el cual la información del comportamiento del parche de una entrada de parche seleccionada contiene información utilizable para modificar el comportamiento de la porción distintiva de

la entidad de programación distintiva.

[c73] 31. El método de la reivindicación 21, por el cual la información del comportamiento del parche de una entrada de parche seleccionada contiene una prueba de parámetro de validación a ser aplicado a un parámetro específico asociado con la entidad de programación distintiva.

[c74] 32. El método de la reivindicación 21, por el cual la identidad de las memorias del computador que guardan la estructura de información de la tabla de parches es configurable.

[c75] 33. El método de la reivindicación 21, por el cual las memorias del computador están directamente conectadas a un sistema de computación en el que una o más entradas de parche serán aplicadas.

[c76] 34. El método de la reivindicación 21, por el cual las memorias del computador están directamente conectadas a un sistema de computación en el que no será aplicada ninguna entrada de parche.

[c77] 35. Un sistema de computación que implementa automáticamente los parches de código recibidos, que comprenden:

una biblioteca preinstalada de funciones de ayuda; y

un agente de parcheo que recibe parches de código, cada uno de los cuales tiene como objetivo un grupo de módulos ejecutables y la identificación de una función de ayuda en la biblioteca, y que, cuando se ejecuta un módulo ejecutable en un grupo objetivo de un parche de código recibido, se invoca la función de ayuda identificada con el parche de código que tiene como objetivo el grupo de módulos ejecutables, incluyendo el módulo ejecutable para realizar la validación de valor.

[c78] 36. El sistema de computación de la reivindicación 35, por el cual el agente de parcheo incluye un subsistema de mantenimiento de la biblioteca que recibe nuevas funciones de validación de parámetro, y agrega automáticamente las nuevas funciones de validación de la biblioteca recibidas, de manera que estas funciones nuevas primarias puedan ser invocadas de acuerdo con los parches de código que los identifican.

[c79] 37. Un método de un sistema de computación para aplicar automáticamente a un parche de software, que comprende:

recepción de un parche en el sistema de computación para modificar el comportamiento de una entidad de programación, el parche (1) que especifica una manera para modificar el comportamiento de una entidad de programación, y (2) para cada una de una variedad de versiones de la entidad de programación, (a) la identificación de la versión de la entidad de programación; (b) la especificación de una posición en la versión de la entidad de programación para modificar el comportamiento de la entidad de programación en la manera especificada, y (c) la identificación del código que se espera resida en la posición especificada en la versión de la entidad de programación antes de que el comportamiento de la entidad de programación sea modificado;

determinación automática de una versión distintiva de la entidad de programación entre las versiones de la entidad de programación identificada por el parche esté instalado en el sistema de computación; y

sólo si la posición especificada para la versión distintiva de la entidad de programación contiene el código identificado que se espera resida en la posición especificada, se modifica el comportamiento de la entidad de programación en la posición especificada para la versión distintiva de la entidad de programación de acuerdo con el parche.

[c80] 38. Un medio reconocido por el computador, cuyo contenido lleve a que un sistema de computación realice un método para aplicar automáticamente un parche de software que incluye:

el almacenamiento de un parche en el sistema de computación para modificar el comportamiento de una entidad de programación, el parche (1) especifica la manera en que se debe modificar el comportamiento de la entidad de programación, y (2) para cada una de una variedad de versiones de la entidad de programación, (a) la identificación de la versión de la entidad de programación; (b) la especificación de una posición en la versión de la entidad de programación para modificar el comportamiento de la entidad de programación en la manera especificada, y (c) la identificación del código que se espera resida en la posición especificada en la versión de la entidad de programación antes de que el comportamiento de la entidad de programación sea modificado; determinando

201

automáticamente de una versión distintiva de la entidad de programación entre las versiones de la entidad de programación identificada por el parche esté instalado en el sistema de computación; y

sólo si la posición especificada para la versión distintiva de la entidad de programación contiene el código identificado que se espera resida en la posición especificada, se modifica el comportamiento de la entidad de programación en la posición especificada para la versión distintiva de la entidad de programación de acuerdo con el parche.

PARCHEO (CONEXIÓN PROVISIONAL) EFICIENTE

Compendio de la Revelación 2

Se describe el dispositivo para procesar automáticamente los parches de software. El dispositivo recibe un paquete de parches distintivos en un sistema de computación para modificar el comportamiento de una entidad de programación. El mismo extrae automáticamente del paquete de parches distintivos (1) información de la aplicación del parche que identifica una entidad de programación distintiva a la cual se aplican los parches, y (2) la información del comportamiento del parche que especifica la manera en que debe modificar el comportamiento de la entidad de programación distintiva. El dispositivo agrega automáticamente una entrada distintiva a una tabla de parche que contiene la información extraída de la aplicación del parche, así como información del comportamiento del parche.

203

TERCERA REIVINDICACIÓN

Reivindicamos:

[c81] 1. Un método en un sistema de computación para aplicar un parche de software utilizando un agente automatizado de parcheo, que incluye:
recepción del parche de software;
como respuesta a la recepción del parche de software sin intervención del usuario:
identificación de una instancia de módulo ejecutable que se encuentra actualmente instalado, al cual corresponde el parche de software recibido; y
la aplicación del parche de software recibido a la instancia del módulo ejecutable que se encuentra actualmente instalado para modificar el comportamiento de la instancia de módulo ejecutable identificada.

[c82] 2. El método de la reivindicación 1, por el cual el módulo ejecutable identificado es un módulo ejecutable de manera independiente.

[c83] 3. El método de la reivindicación 1, por el cual el módulo ejecutable identificado es una biblioteca.

[c84] 4. El método de la reivindicación 1, por el cual el módulo ejecutable identificado es un programa.

[c85] 5. El método de la reivindicación 1, por el cual el módulo ejecutable identificado es un módulo de código administrado.

[c86] 6. El método de la reivindicación 1, por el cual dos parches de software se reciben como correspondientes a la misma instancia de módulo ejecutable, y ambos parches de software recibidos se aplican para modificar el comportamiento de la instancia de módulo ejecutable.

[c87] 7. El método de la reivindicación 6, por el cual dos parches de software recibidos corresponden a ubicaciones diferentes en la instancia del módulo ejecutable.

[c88] 8. El método de la reivindicación 6, por el cual dos parches de software recibidos corresponden a las mismas ubicaciones en la instancia del módulo ejecutable.

[c89] 9. El método de la reivindicación 8, por el cual los parches recibidos se aplican de manera que las dos modificaciones de comportamiento especificadas en los parches, estén exhibidos por la instancia del módulo ejecutable parcheado.

[c90] 10. El método de la reivindicación 8, por el cual los parches recibidos se aplican de manera que sólo la modificación del comportamiento del dominante entre los dos parches de software recibidos, estén exhibidos por la instancia del módulo ejecutable.

[c91] 11. El método de la reivindicación 10, por el cual el parche de software dominante recibido contiene una indicación explícita que es para dominar uno o más parches de software.

[c92] 12. El método de la reivindicación 1, por el cual la instancia de módulo ejecutable identificada exhibe el comportamiento modificado en un momento posterior a la aplicación del parche de software.

[c93] 13. El método de la reivindicación 1, que además incluye, utilizando el agente automatizado de parcheo, la aplicación del parche de software recibido a una imagen de disco para la instancia de módulo ejecutable identificada.

[c94] 14. El método de la reivindicación 1, que además incluye, utilizando el agente automatizado de parcheo, la aplicación del parche de software recibido a la instancia de módulo ejecutable identificada cuando dicha la instancia de módulo ejecutable identificada se reinstale en un momento posterior.

[c95] 15. El método de la reivindicación 1, por el que la instancia de módulo ejecutable actualmente instalada, a la cual corresponde el parche de software, se identifica de entre una variedad de instancias de módulo ejecutables al cual corresponde el parche de software, y por el que el parche de software recibido se aplica utilizando la información específica de instancia de módulo para la instancia de módulo ejecutable identificada incluida en el parche recibido.

[c96] 16. El método de la reivindicación 15, por el cual la variedad de instancias de módulo ejecutables son cada una de éstas una versión diferente del mismo módulo ejecutable.

[c97] 17. El método de la reivindicación 15, por el cual la información del módulo específica a la instancia para el módulo ejecutable identificado, contenido en el

205

parche recibido, especifica una desviación dentro del módulo ejecutable identificado en el cual se modifica el comportamiento del módulo ejecutable identificado.

[c98] 18. El método de la reivindicación 17, por el cual la información del módulo específica a la instancia para el módulo ejecutable identificado contenido en el parche recibido, especifica además el contenido de la instancia de módulo ejecutable identificado que se espera aparezca en una desviación específica antes de aplicar el parche recibido.

[c99] 19. El método de la reivindicación 1, que además incluye, utilizando el agente automatizado de parcheo:

la recepción de un segundo parche de software que corresponde a un módulo ejecutable que no está presente en el sistema de computación; y

como respuesta a la recepción del segundo parche de software, sin la intervención del usuario, el almacenamiento del parche de software recibido para una futura aplicación si una instancia de módulo ejecutable a la que le corresponde el parche de software recibido se instala al sistema más adelante.

[c100] 20. El método de la reivindicación 1, en el que, luego de la aplicación, se desinstala el módulo ejecutable identificado y se reinstala, el método incluye además, la utilización del agente automatizado de parcheo sin la intervención del usuario, reaplicando el parche de software recibido a la instancia del módulo ejecutable identificado para modificar el comportamiento de la instancia del módulo ejecutable identificado reinstalado.

[c101] 21. El método de la reivindicación 1, por el que el parche de software recibido contiene código que especifica cómo modificar el comportamiento de la instancia de módulo ejecutable identificado.

[c102] 22. El método de la reivindicación 1 por el que el parche de software recibido contiene información que especifica cómo modificar el comportamiento de la instancia de módulo ejecutable identificado.

[c103] 23. El método de la reivindicación 1 por el que el parche de software recibido especifica la modificación del comportamiento de la instancia de módulo ejecutable identificado a agregar una prueba y los resultados de la misma que

206

involucran lógica a una porción asignada de la instancia de módulo ejecutable identificado.

[c104] 24. El método de la reivindicación 1 por el que el parche de software recibido especifica la modificación del comportamiento de la instancia de módulo ejecutable identificado a agregar una prueba y los resultados de la misma que involucran lógica a una porción asignada de la instancia de módulo ejecutable identificado.

[c105] 25. El método de la reivindicación 1 por el que el parche de software recibido especifica la modificación del comportamiento de la instancia de módulo ejecutable identificado a agregar una función de validación de parámetro a una función asignada en la instancia de módulo ejecutable identificado, por medio de la identificación de una función de validación de parámetro, que será llamada parte de la ejecución de dicha función asignada.

[c106] 26. El método de la reivindicación 1, por el que el comportamiento modificado de la instancia de módulo ejecutable identificado puede ser activado o desactivado con la utilización de una interfase de configuración de parche.

[c107] 27. El método de la reivindicación 1, por el que la utilización del comportamiento modificado de la instancia de módulo ejecutable identificado puede ser activada o desactivada con la utilización de una interfase de configuración de parche.

[c108] 28. El método de la reivindicación 1, por el que el parche de software se recibe al permitir que dicho parche de software sea copiado a una ubicación predeterminada, y por el que la identificación y la aplicación se realiza automáticamente como respuesta a la copia de dichos parche de software a dicha ubicación predeterminada.

[c109] 29. Un medio reconocido por el computador, cuyo contenido genere que un sistema de computación realice un método para aplicar un parche de software, que incluye, la utilización de un agente automatizado de parcheo a

la recepción del parche de software;

como respuesta a la recepción del parche de software:

la identificación de una instancia de módulo ejecutable correspondiente a dicho

parche de software; y

la aplicación del parche de software recibido a la instancia de módulo ejecutable actualmente en utilización para modificar el comportamiento de dicha instancia de módulo ejecutable.

[c110] 30. El medio reconocido por el computador de la reivindicación 29, por el que dicho medio reconocido por el computador es un medio de almacenamiento de información.

[c111] 31. El medio reconocido por el computador de la reivindicación 29, por el que dicho medio reconocido por el computador es un medio de transmisión de información.

[c112] 32. Un sistema de computación para aplicar el parche de software, que incluye:

un receptor que recibe automáticamente el parche de software;

un subsistema de identificación, que como respuesta recibida por parte del receptor del parche de software, sin la intervención de usuario, identifica una instancia de módulo ejecutable actualmente instalada correspondiente al parche de software recibido; y un subsistema de aplicación de parche, que, como respuesta a la identificación de una instancia de módulo ejecutable actualmente instalada correspondiente a al parche de software recibido por medio del subsistema de identificación, sin la intervención de usuario, aplica el parche de software recibido a dicha instancia de módulo ejecutable actualmente instalada para modificar el comportamiento de dicha instancia de módulo ejecutable identificado.

[c113] 33. Un medio reconocido por el computador que contiene una estructura de información de parche de software que representa un parche de software, por el cual dicha estructura de información de parche de software incluye:

la primera información que identifica uno o más módulos ejecutables de software a los cuales se aplica el parche; y la segunda información que especifica la manera en que dichos módulos ejecutables de software deben ser modificados de manera que el contenido de dicha estructura de información de parche de software pueda ser utilizada por un agente automático de parcheo,

208

seleccionar un módulo ejecutable de software identificado por la primera información y modificar el módulo ejecutable de software seleccionado en la manera especificada por la segunda información.

[c114] 34. El medio reconocido por el computador de la reivindicación 33, por el que dicho medio reconocido por el computador es un medio de almacenamiento de información.

[c115] 35. El medio reconocido por el computador de la reivindicación 33, por el que dicho medio reconocido por el computador es un medio de transmisión de información.

[c116] 36. El medio reconocido por el computador de la reivindicación 33, por el que la segunda información es un código que especifica la manera en que los módulos ejecutables de software identificados deben ser modificados.

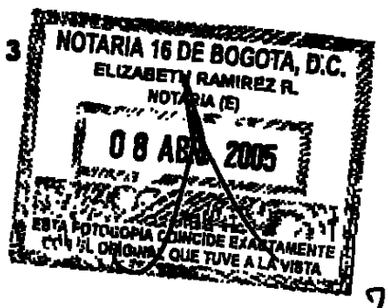
[c117] 37. El medio reconocido por el computador de la reivindicación 33, por el que la segunda información es información que indica la manera en que los módulos ejecutables de software identificados deben ser modificados.

[c118] 38. El medio reconocido por el computador de la reivindicación 33, por el que la estructura de información de parche de software incluye además una firma que demuestra que dicha estructura de información de parche de software fue generada por una autoridad de parche asignada y que no ha sido modificada desde la creación de la firma.

[c119] 39. El medio reconocido por el computador de la reivindicación 38, por el que la identidad de un firmante puede ser distinguido de la firma de la estructura de información de parche de software, y dicha identidad de firmante distinguida puede ser comparada con una identidad de firmante que puede ser distinguida de la firma del módulo ejecutable de software identificado.

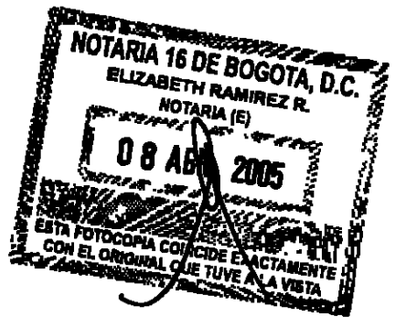
PARCHEO EFICIENTE

COMPENDIO DE LA REVELACIÓN 3



209

Se describe el dispositivo para aplicar los parches de software. Con la utilización de agente de parcheo automático, el dispositivo recibe el parche de software. Como respuesta a la recepción del mismo, el dispositivo, sin la intervención del usuario, realiza lo siguiente: Primero, el dispositivo identifica una instancia de un módulo ejecutable que se encuentra actualmente instalado, y al cual le corresponde el parche de software recibido. Segundo, el dispositivo aplica el parche de software recibido a la instancia de módulo ejecutable identificado que se encuentra actualmente instalado para modificar el comportamiento de dicha instancia de módulo ejecutable identificado.



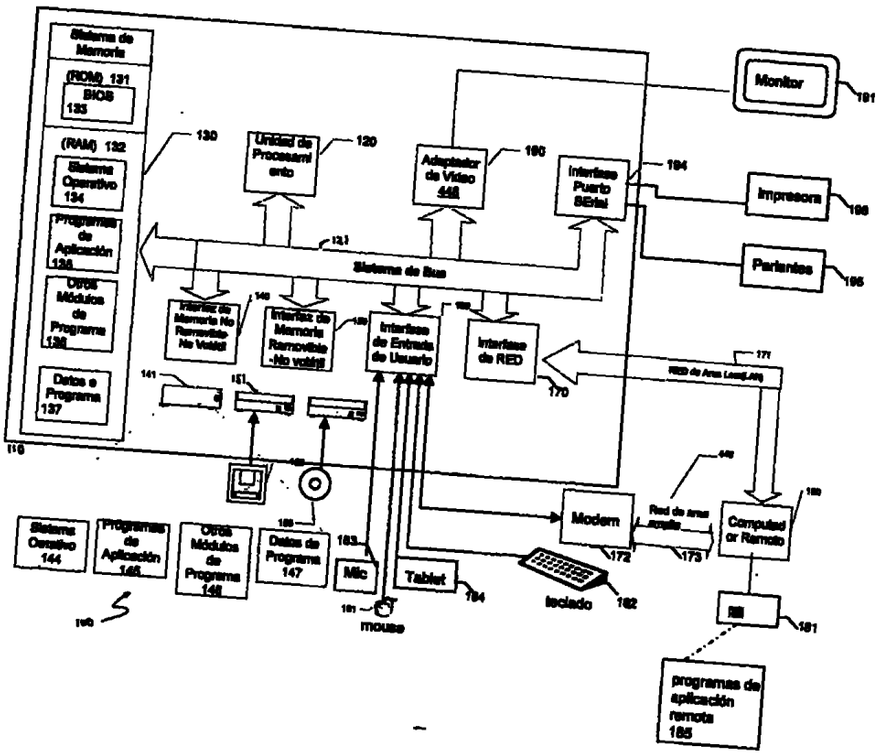


Fig. 1

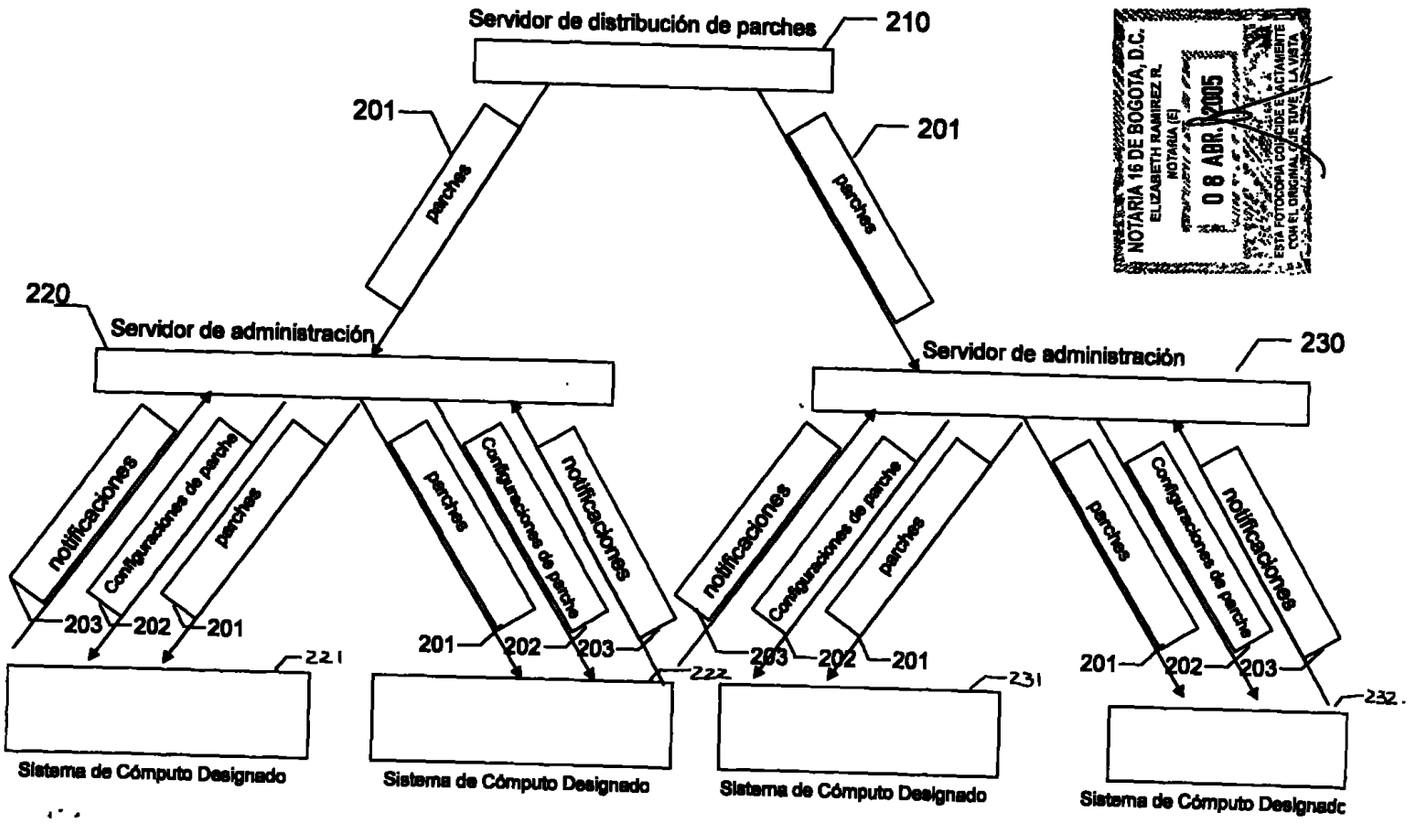


Fig. 2

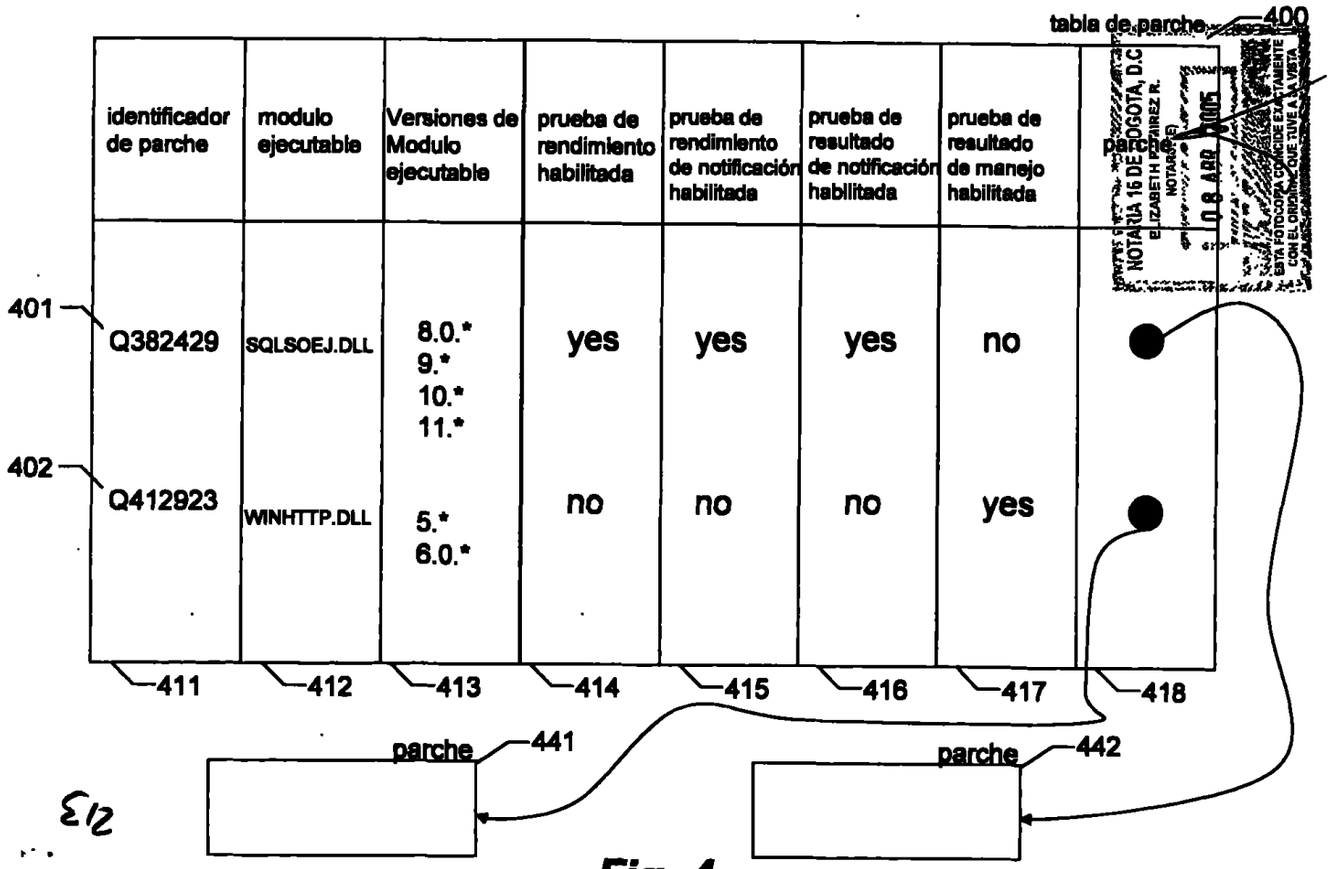


tabla de parche
400
identificador de parche
modulo ejecutable
Versiones de Modulo ejecutable
prueba de rendimiento habilitada
prueba de rendimiento de notificación habilitada
prueba de resultado de notificación habilitada
prueba de resultado de manejo habilitada
parche
401
Q382429
SQLSOEJ.DLL
8.0.*
9.*
10.*
11.*
yes
yes
yes
no
402
Q412923
WINHTTP.DLL
5.*
6.0.*
no
no
no
yes
411
412
413
414
415
416
417
418
parche
441
parche
442

Fig. 4

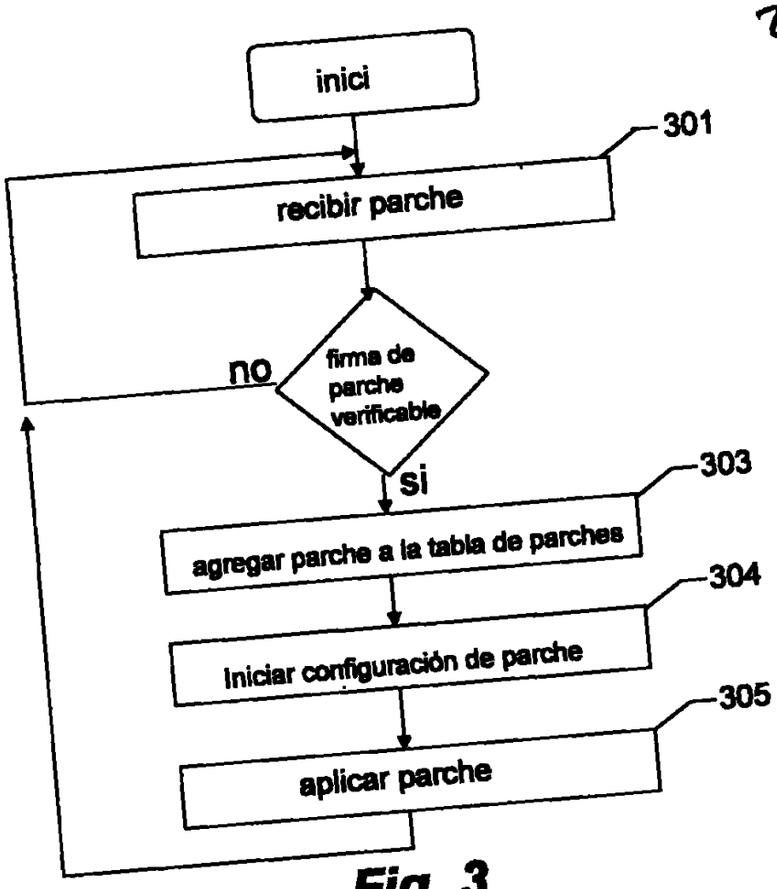


Fig. 3

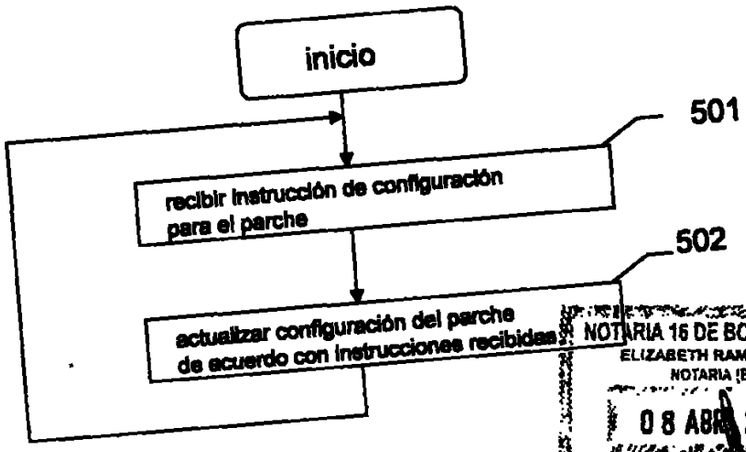


Fig. 5

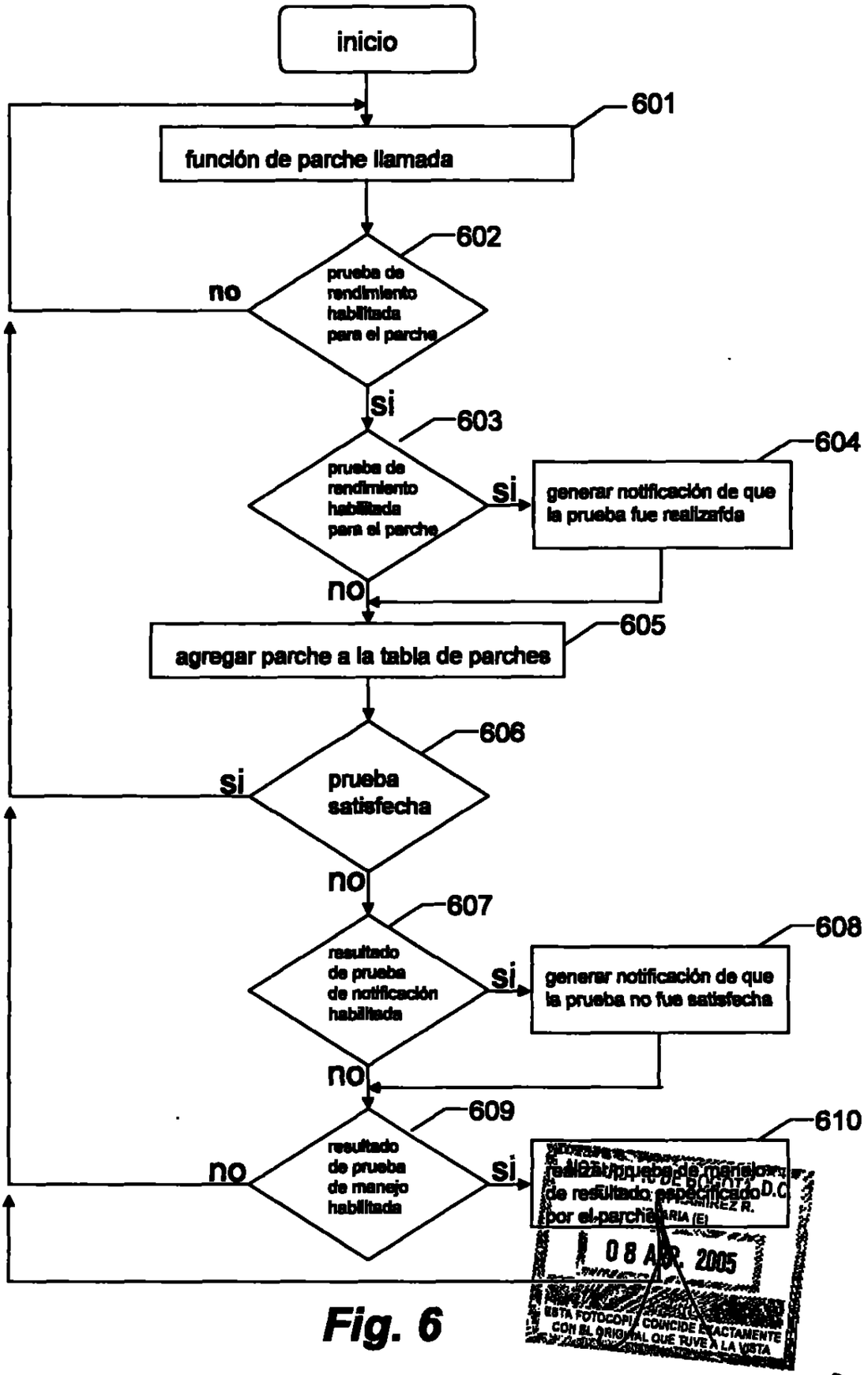

inicio
601
función de parche llamada
602
prueba de rendimiento habilitada para el parche
no
si
603
prueba de rendimiento habilitada para el parche
si
604
generar notificación de que la prueba fue realizada
no
605
agregar parche a la tabla de parches
606
prueba satisfecha
si
no
607
resultado de prueba de notificación habilitada
si
608
generar notificación de que la prueba no fue satisfecha
no
609
resultado de prueba de manejo habilitada
no
si
610
realizar prueba de manejo de resultado especificado por el parche
08 ABR. 2005
ESTA FOTOCOPIA COINCIDE EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

Fig. 6

2

216

No. 1243227

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

November 02, 2004

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: ***60/570,124***
FILING DATE: ***May 11, 2004***

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

NOTARIA BOGOTA, D.C.
ELIZABETH RAMIREZ
NOTARIA (E)
09 APR. 2005
ESTA FOTOCOPIA CORRESPONDE EXACTAMENTE CON EL ORIGINAL QUE TUVE A LA VISTA

L. EBELEN
Certifying Officer

2 of 6

217

PTO/SB/16 (08-03)
Approved for use through 07/31/2006. OMB 0651-0032
Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53 (c).

Express Mail Label No. EV336678673US

INVENTOR(S)		
Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)

☐ Additional inventors are being named on the _____ separately numbered sheets attached hereto

TITLE OF THE INVENTION (500 characters max)

EFFICIENT PATCHING

CORRESPONDENCE ADDRESS

Direct all correspondence to:
☒ Customer Number 25096

OR

☒ Firm or Individual Name Perkins Coie LLP

ENCLOSED APPLICATION PARTS (check all that apply)

☒ Specification Number of Pages 49 ☐ CD(s), Number

☒ Drawing(s) Number of Sheets 5 ☒ Other (specify) Postcard

☐ Application Data Sheet. See 37 CFR 1.76

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT

☐ Applicant claims small entity status. See 37 CFR 1.27.

☐ A check or money order is enclosed to cover the filing fees — FILING FEE AMOUNT ($)

☐ The Commissioner is hereby authorized to charge any additional filing fees or credit any overpayment to Deposit Account Number: 50-0665

☐ Payment by credit card. Form PTO-2038 is attached.

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No.

☐ Yes, the name of the U.S. Government agency and the Government Contract number are: _____.

Respectfully submitted,

SIGNATURE _____ Date May 11, 2004

TYPED or PRINTED NAME Steven D. Lawrenz REGISTRATION NO. (if appropriate) 37,376

TELEPHONE (206) 359-8000 Docket Number: 41626.801DUS1

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

27

EXPRESS MAIL NO. EV338675073US

EFFICIENT PATCHING

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is related to U.S. Patent Application No. 10/307,902, entitled "PATCHING OF IN-USE FUNCTIONS ON A RUNNING COMPUTER SYSTEM," and filed on December 2, 2002; U.S. Patent Application No. _____, (patent counsel's docket number _____), entitled "EFFICIENT PATCHING", filed concurrently herewith; and U.S. Patent Application No. _____, (patent counsel's docket number _____), entitled "EFFICIENT PATCHING", filed concurrently herewith, each of which is hereby incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] The present invention is directed to the field of updating the operation of installed computer programs.

BACKGROUND

[0003] Patching is the process of modifying already-installed programs, including application programs, utility programs, operating systems and operating system components, device drivers, etc. Patching can be useful to modify programs for a variety of purposes, including correcting a programming error, reducing or eliminating a security risk, or improving the logic used by the modified program. Patching is typically initiated by the company or other organization that originally supplied the program to be patched.

[0004] Installed programs are predominantly composed of executable code modules. As one example, many programs designed to execute on the WINDOWS XP operating system from Microsoft Corp. of Redmond, Washington

are predominantly composed of executable code modules called "DLLs." One popular conventional approach to patching is to identify, among the executable code modules making up the installed program to be patched, the executable code module containing the program code that one wishes to modify with a patch; creating a new version of the identified executable code module in which the desired modification is made; and distributing the new version of the identified executable code module, together with an installer program, to users who may wish to apply the patch. Each user then determines whether s/he wishes to apply the patch, and if so, executes the installer program, which replaces the original version of the identified executable code module with the new version of the identified executable code module.

[0005] Conventional approaches to patching have a number of significant disadvantages. These disadvantages often increase the burden associated with receiving and applying patches. In some cases, this increased burden delays the application of some patches by some users, and even prevents the application of some patches by some users. Such delay and prevention in the application of patches can in some cases have serious negative consequences for users, especially for patches designed to reduce or eliminate a security risk.

[0006] One disadvantage of conventional approaches to patching relates to common cases in which multiple patches must be created in distributed to effect a single modification to a single program. In some cases, the program to be patched has several different "flavors" of a particular executable code module, such as a different flavor for each operating system or operating system version on which the program is designed to execute, and/or each natural language version of the program. Where the identified executable code module is such an executable code module, the patch creation and distribution process described above must be repeated for each flavor of the identified executable code module. The user must then select and apply the patch for the appropriate flavor of the identified executable code module. Sorting through the resulting large number of patches and selecting the proper set of patches for application on each user's

computer system can be very burdensome, so much so that this condition is sometimes called "patch hell." In some cases, an administrator must maintain an inventory database identifying the set of executable module versions installed on each target system, which is used to select appropriate conventional patches for each target system.

[0007] Another disadvantage of conventional approaches to patching relates to the large size of the distributed patches. It is not uncommon for executable code modules to have a size measured in megabytes, which in turn causes a single patch to have a comparable size, making it unwieldy to distribute and store, or even impossible to distribute and store, for some users. This problem can be multiplied for patches having multiple flavors. Further, because each conventional patch typically includes an entire substitute executable module, applying conventional patches can contribute to the problem of code churn.

[0008] A further disadvantage of conventional approaches to patching relates to a need by some users to test patches before applying them to production computer systems. In some cases, installing a patch on a computer system can have adverse results, such as where the new version of the identified executable code module contained in the patch introduces a new programming error, or where it causes a new, unpredicted interaction with another program running on the computer system against which it is applied. Accordingly, often before applying a patch against a production system whose data and operation are important to sustain, the user first applies the patch against a testing system to evaluate whether the patch is safe to apply to the production system. Such separate testing of patches adds to the burden associated with patching. Additionally, where a conventional patch creates a problem – such as an application compatibility problem or a new exploit vulnerability – at a time substantially after the patch is applied, it can be difficult to trace such problems back to the patch.

[0009] An additional disadvantage of conventional approaches to patching relates to the operation of the installer included in the patch. Often, in order to replace an executable code module that is part of executing program, the installer must first

terminate execution of that program. Also, in some cases, such replacement cannot be completed without restarting the computer system. Both of these steps can cause substantial disruptions in the use of the patched computer system.

[0010] Another disadvantage of conventional approaches to patching involves attempting to patch an executable module for which a "private fix," also called a "hot fix," has earlier been issued to a proper subset of the customers for that executable module. In such cases, because of difficulties encountered in distributing conventional patches that substitute different new versions of the executable code module depending upon to each user depending upon whether the user has applied the hot fix, it is typical to instead distribute a simple conventional patch that substitute a single new version of the executable module irrespective of whether the user has applied the hot fix. If that new version embodies the hot fix, the patch imposes the hot fix on customers not intended to receive it. If, on the other hand, that new version does not embody the hot fix, it deprives the customers intended to receive the hot fix of the hot fix.

[0011] Another disadvantage of conventional approaches to patching involves the fact that the installers for software products that rely on a particular executable module, such as a particular dynamic link library, often "hide" that executable module by storing it in a non-standard location in the target computer system's filesystem. Accordingly, it is sometimes difficult or impossible to determine whether a particular target system contains a copy of an executable module to be patched, and, if so, where it resides in the target computer system's filesystem. Also, some software products maintain a "catalog" of executable module versions that have been installed by their installers. A software product may rely upon the correctness of an indication in the catalog of the version of a particular executable module. Such reliance is defeated where a conventional patch replaces the version of the executable module identified in the catalog with a new version of the executable module without updating the catalog.

[0012] Another disadvantage of conventional approaches to patching springs from the fact that they are impossible to apply at a time before the executable module

to be patched is installed in the target computer system. As a result, if the executable module to be patched is installed in the target computer system after a conventional patch for that executable module is received, it is unlikely that the patch will be applied to the executable module.

[0013] Another disadvantage of conventional approaches to patching is that they typically can only be applied by a user logged in to the target computer system using an administrative account having liberal modification permissions. Logging into an administrative account for this purpose can make the target computer system vulnerable to viruses present on the target computer system seeking to modify aspects of the target computer system and requiring liberal permissions to do so.

[0014] Another disadvantage of conventional approaches to patching is that conventional patches can be difficult or impossible to disable, requiring steps such as reversing the replacement of an executable module, or reversing one or more modifications to the system registry.

[0015] Accordingly, a new approach to patching that overcame some or all of the disadvantages of conventional approaches to patching discussed above would have significant utility.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] Figure 1 illustrates an example of a suitable computing system environment in which the facility may be implemented.

[0017] Figure 2 is a data flow diagram showing a typical exchange of data between computer systems in accordance with the facility.

[0018] Figure 3 is a flow diagram showing steps typically performed by the facility in order to receive and process a new patch.

[0019] Figure 4 is a data structure diagram showing a sample patch table typical of those used by the facility.

[0020] Figure 5 is a flow diagram showing steps typically performed by the facility in order to update configuration instructions for a particular patch.

[0021] Figure 6 is a flow diagram showing steps typically performed by the facility to perform the parameter validation specified by a patch.

DETAILED DESCRIPTION

[0022] A software facility for patching installed computer program code ("the facility") is provided. In some embodiments, the facility adds parameter testing and test result handling to installed functions. In other embodiments, the facility adds various other kinds of functionality to installed functions, in some cases at arbitrary positions in the installed functions' flows of execution.

[0023] In some embodiments, for each patch, the facility distributes to each computer system to be patched -- i.e., each "target computer system" -- the specification of a point at which to perform a test, the identity of the test to perform, and how to act in response to one or more different test results. In some embodiments, the facility provides a standard set of parameter validation and other tests whose use can be specified in patches. For example, a patch may specify that, for a particular function, if a particular parameter of the function does not have a certain value, invocation of the function should fail before its substantive execution begins. Another patch may specify that, for a particular function, if a particular parameter has a length exceeding a specified maximum length, the parameter should be truncated to the specified maximum length before execution of the function is permitted to proceed. Many security exploits rely on causing functions to be called with parameter values that, while they were not blocked in the original version of a function's code, cause the function to create or take advantage of an unsafe condition. In many cases, such exploits can be prevented by using such patches to prevent the function from executing with such parameter values. In some embodiments, the patches specify the testing of values other than function parameters, such as values read from a file or inputted by user.

[0024] In some embodiments, an automated patching agent automatically receives each patch, validates it, and stores it in a patch table for possible application. In

some embodiments, each patch is applied to any instances of the executable module to be patched that are already loaded on the target computer system when the patch is received. This approach is referred to herein as "hot patching," and enables patches to become effective immediately upon being received, and does not require the facility to be able to determine where the executable module to be patched is stored on disk. In some embodiments, each received patch is applied to the disk image of the executable module to be patched, so that, when the disk image is loaded a future times, the loaded disk image includes the patch. This approach is referred to herein as "cold patching," and permits patches to be persistent across multiple sessions. In some embodiments, the facility performs both hot and cold patching. In some embodiments, each patch is applied to the executable module to be patched each time the executable module to be patched is loaded by the operating system's loader. This approach is referred to herein as "load-time patching." In some embodiments, each patch is applied to the executable module to be patched each time the function to be patched is invoked. This approach is referred to herein as "call-interception patching." Load-time patching call-interception patching both (1) do not require the facility to be able to determine where the executable module to be patched is stored on disk, (2) facilitate ready reversibility of a particular patch, and (3) do not require modification of the executable module's disk image.

[0025] In some embodiments, the facility permits a user or administrator to configure the operation of patches that have been applied. As examples, such configuration can include, for a particular applied patch: whether the test specified by the patch is performed when execution reaches the point specified for the patch; whether the test result handling specified by the patch is performed or ignored; and/or whether performance of the test and/or its result is logged, displayed in warning message, etc. In these embodiments, the facility permits patches to be tested on production computer systems by initially enabling logging and disabling result handling. In these embodiments, the facility further permits operation of the patch to be logged in a "verbose mode" after enabling its result

handling to assist with identifying instances in which the patch is creating a problem, such as an application compatibility problem or other IT problem. These embodiments also permit a patch to be quickly disabled after being applied if it is discovered that the patch is creating problems. Some embodiments of the facility also permit the patch to be quickly disabled by simply deleting the patch from the set of patches received and stored in the target computer system.

[0026] In some embodiments, the facility uses a "data-driven" patching approach, in which patches contain no code, but rather data, such as a small human-readable text or XML document, that specifies a point at which to perform a test, the identity of the test to perform, and how to act in response to one or more different test results. In such embodiments, a patching agent receives data-driven patches, and adds the tests and test handling specified by the patches. In some embodiments, the facility uses a "code-driven" patching approach, and which each patch contains a short program to be added to the executable module to be patched that itself performs the test by calling the facility's standard parameter testing functions, and that itself performs the test handling. Using data-driven patching or code-driven patching, it is sometimes possible to address all flavors of an executable module to be patched with a single patch.

[0027] In some embodiments, the facility signs each patch to demonstrate both that (1) the patch is from an approved source, and (2) the contents of the patch have not been modified since the time at which the patch was created by the approved source.

[0028] In some embodiments, the facility distributes every patch to every target computer system, and a patching agent on the target computer system determines automatically which patches or to be applied on the target computer system, and how they are to be applied, based upon the target computer system's characteristics. This relieves users and administrators of many of the burdens conventionally associated with selecting and applying patches, and the burden of maintaining an accurate and current inventory database. For example, these characteristics can include which version of the executable module to be patched

is installed on the target computer system. In these embodiments, the facility can overcome the type of problems typically caused by hot fixes, by distributing patches that specify different handling for hot-fixed and un-hot-fixed flavors of a particular executable module, obviating any need to either sacrifice a hot fix for a particular executable module or make the hot fix ubiquitous when patching that executable module.

[0029] In some embodiments, the patching agent stores every patch received in the target system, irrespective of whether the executable module to be patched by a particular patch is installed on the target system when that patch is received. Because the facility in many cases applies patches in response to the loading of an executable module to be patched or the invocation of a function to be patched, the facility can apply a patch to an executable module that was installed on the target system after that patch was received in the target system. Also, patches can survive the uninstallation and subsequent reinstallation of the executable module to be patched.

[0030] In some embodiments, the patching agent is implemented in an operating system service. In these embodiments, the facility accords the patching agent any permissions required to apply a patch. These embodiments reduce the security risk typically imposed when conventional patches are applied, as they obviate any need for a user to log in to the target computer system using an administrative account having broad modification permissions, and thereby give any viruses present on the target computer system a greater opportunity to modify sensitive aspects of the target computer system.

[0031] The patches used by the facility are typically relatively small, and therefore impose modest resource requirements for transmission and storage. Also, because the patches used by the facility tend to modify the behavior of the patched software in a small number of well-defined ways, the facility helps to reduce the problem of code churn.

[0032] Figure 1 illustrates an example of a suitable computing system environment 100 in which the facility may be implemented. The computing system

environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the facility. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0033] The facility is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the facility include, but are not limited to: personal computers, server computers, hand-held or laptop devices, tablet devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0034] The facility may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, and so forth, which perform particular tasks or implement particular abstract data types. The facility may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in local and/or remote computer storage media including memory storage devices.

[0035] With reference to Figure 1, an exemplary system for implementing the facility includes a general purpose computing device in the form of a computer 110. Components of the computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any

of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0036] The computer 110 typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by the computer 110 and includes both volatile and nonvolatile media, and removable and non-removable media. By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can accessed by the computer 110. Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer-readable media.

[0037] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and

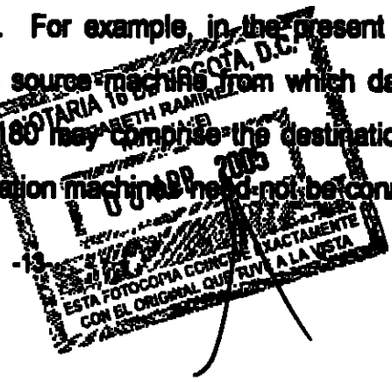
random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Figure 1 illustrates operating system 134, application programs 135, other program modules 136 and program data 137.

[0038] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Figure 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0039] The drives and their associated computer storage media, discussed above and illustrated in Figure 1, provide storage of computer-readable instructions, data structures, program modules and other data for the computer 110. In Figure 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146 and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers herein to

illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 110 through input devices such as a tablet, or electronic digitizer, 164, a microphone 163, a keyboard 162 and pointing device 161, commonly referred to as mouse, trackball or touch pad. Other input devices not shown in Figure 1 may include a joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. The monitor 191 may also be integrated with a touch-screen panel or the like. Note that the monitor and/or touch screen panel can be physically coupled to a housing in which the computing device 110 is incorporated, such as in a tablet-type personal computer. In addition, computers such as the computing device 110 may also include other peripheral output devices such as speakers 195 and printer 196, which may be connected through an output peripheral interface 194 or the like.

[0040] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet. For example, in the present facility, the computer system 110 may comprise source machine from which data is being migrated, and the remote computer 180 may comprise the destination machine. Note however that source and destination machines need not be connected by a



network or any other means, but instead, data may be migrated via any media capable of being written by the source platform and read by the destination platform or platforms.

[0041] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160 or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Figure 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0042] While various functionalities and data are shown in Figure 1 as residing on particular computer systems that are arranged in a particular way, those skilled in the art will appreciate that such functionalities and data may be distributed in various other ways across computer systems in different arrangements. While computer systems configured as described above are typically used to support the operation of the facility, one of ordinary skill in the art will appreciate that the facility may be implemented using devices of various types and configurations, and having various components.

[0043] Figure 2 is a data flow diagram showing a typical exchange of data between computer systems in accordance with the facility. The computer systems shown in Figure 2 (computer systems 210, 220, 221, 222, 230, 231, and 232) typically have some or all of the components shown and discussed in conjunction with Figure 1. At the patch distribution server, the facility generates one or more patches. These patches 201 are sent from the patch distribution server to one or

more administration servers, such as administration servers 220 and 230. Each administration server, in turn, forwards the patches to one or more target computer systems, such as target computer systems 221, 222, 231, and 232. In some embodiments (not shown), the patch distribution server sends patches directly to one or more target computer systems, or via a more indirect route than through a single administration server. Patches received at a target computer system are processed at the target computer system as described in greater detail below. An administration server can also send patch configuration commands 202 to one or more target computer systems, which apply the patch configuration commands to reconfigure the operation of particular patches. As is discussed in greater detail below, a patch may be completely disabled; if a patch is not disabled, notification of its operation and its test result handling may each be independently enabled or disabled. When notification of its operation is enabled, notifications may be displayed or stored locally on the target computer system, or may be sent as notifications 203 to an appropriate administration server.

[0044] Figure 3 is a flow diagram showing steps typically performed by the facility in order to receive and process a new patch. In step 301, the facility receives a patch. The patch received in step 301 may be either a data-driven patch or a code-driven patch. A sample data-driven patch is shown in Table 1 below.

```
1     <Softpatch Patch="Q382429">
2      <AffectedApplication AffectedExe="sqlservr.exe">
3       <AffectedVersion Version="9.*">
4        <AffectedModules Name="SQLSORT.DLL">
5         <Version "8.0.*, 9.*">
6              <Function Name="SsrpEnumCore" Address="0x0802E768"
7                  Param="2" Paramtype="LPSTR">
8               <Filter MaxByteLength="60" />
9               <Resolution ActionType="BOOL" Action="FALSE" />
10             </Function>
11        </Version>
12        <Version "10.*, 11.*">
13             <Function Name="SsrpEnumCore" Address="0x0802D283"
14                 Param="2" Paramtype="LPSTR">
15              <Filter MaxByteLength="128" />
16              <Resolution ActionType="BOOL" Action="FALSE" />
17             </Function>
18        </Version>
19       </AffectedModules>
20      </AffectedVersion>
21     </AffectedApplication>
22
23     <Signature Hash="MD5" Signature="C509-64AA-9161-8C52-
24      9F6D-BF5A-AEF2-ECE1-0038-34D1"/>
25    </Softpatch>
```

TABLE 1

[0045] Line 1 contains a unique identifier for the patch. Line 2 identifies the application affected by the patch. Line 3 identifies the application version affected by the patch. Line 4 identifies the executable module affected by the patch. Line 5 identifies two versions of the affected executable module -- versions 8.0 .* and 9.* -- for which patching directions are provided. Lines 6-10 contain patching directions for these two versions of the executable module. Lines 6-7 identify the function to be patched, its address in the executable module, its parameter to be tested by the patch, and the type of the parameter to be tested. Line 8 indicates that the parameter identified in lines 6-7 should be tested to determine whether its length exceeds 60 bytes. Line 9 indicates that the invocation of the function should fail if the test succeeds. Line 12 identifies two more versions of the affected executable module -- versions 10.* and 11.*-- for which patching directions are provided. Lines 13-17 contain patching directions for these two versions of the executable module. Lines 13-14 identify the function to be patched, its address in the executable module, its parameter to be tested by the

patch, and the type of the parameter to be tested. It can be seen that the address of the function to be patched in the executable module identified in lines 13-14 for versions 10 .*and 11 .* is different from the address of the function to be patched identified in lines 6-7 for versions 8.0 .* and 9.*. Line 15 indicates that the parameter identified in lines 13-14 should be tested to determine whether its length exceeds 128 bytes. Line 16 indicates that the invocation of the function should fail if the test succeeds. The patch may specify a variety of result handling action types, including failing the invocation of the patched function, raising an exception, terminating the process in which the patched executable module is being executed, or correcting the offending value (such as by truncating an overlong string). Lines 23-25 contain a signature for the patch that both identifies the source of the patch and verifies that the patch has not been altered since leaving its source.

[0046] Table 2 below contains a code-driven version of the patch shown in Table 1 above.

```
1     00411A7E  push     3Ch
2     00411A80  mov      eax,dword ptr [str]
3     00411A83  push     eax
4     00411A84  call     ValidateStringLength (411082h)
5     00411A89  add      esp,8
6     00411A8C  movzx    ecx,al
7     00411A8F  test     ecx,ecx
8     00411A91  je       411A9Ah
9     00411A93  jmp      foo+2 (411AD2h)
10    00411A9A  xor      eax,eax
11    00411A9C  ret
```

TABLE 2

[0047] Lines 1-3 push parameters for the testing function onto the stack. Line 4 calls the testing function. Lines 5-8 branch on the return code for the testing function. If the testing function succeeded, line 9 jumps back to begin executing the body of the patched function. If the testing function failed, lines 10-11 push a failure result code onto the stack and returns from the patched function to the color of the patched function. For readability, Table 2 omits certain details present in some code-driven patches, including the verifiable signature, instructions

that test for the current values of the patch configuration flags, and instructions relocated from the beginning of the code for the patched function.

[0048] In some embodiments, patches of both types may contain additional information, including, for each of one or more versions of the executable module to be patched, a file signature that can be used to validate that a particular instance of the executable module is a proper copy of that version. Such a file signature may be, for example, a size or checksum for the entire executable module version, or the code expected to occur at a particular point in the executable module, such as at the offset at which the executable module is to be patched.

[0049] In step 302, if the patch is signed with a valid signature, then the facility continues in step 303, else the facility continues in step 301 to receive the next patch. In step 303, the facility adds the patch to a local patch table. In step 304, the facility initializes the initial configuration of the patch, such as by sending it to a default configuration.

[0050] Figure 4 is a data structure diagram showing a sample patch table typical of those used by the facility. The patch table 400 contains rows, such as rows 401 and 402, each divided into the following columns: a patch identifier column 411, containing the patch identifier extracted from the patch; an executable module column 412, containing information identifying the executable module to be patched, such as its name; an executable module versions column 413 identifying all of the versions of the executable module identified in column 412 to which the patch applies; a test performance enabled column 414, containing the current configuration value for whether the test specified by the patch should be performed each time the patched function is called; a test performance notification enabled column 415, containing the current configuration value for whether a notification should be generated each time the patch's test is performed; a test result notification enabled column 416, containing the current configuration value for whether a notification should be generated each time the patch's test succeeds; a test result handling enabled column 417 containing the current

configuration value for whether the patch's result handling should be implemented when the patch's test fails; and a patch column 418 that contains a pointer to the patch itself, specifying a test and test result handling to be performed each time the test fails. In some embodiments, rather than containing a pointer to the patch as shown, the patch column 418 contains each patch directly. A particular patch table may contain or point to patches of various types, such as all code-driven patches, all data-driven patches, or a combination of code-driven and data-driven patches.

[0051] In step 305, once the facility has added the received patch to the patch table and initialized its configurations, the patch is available to be automatically applied by the facility to the executable module. In step 305, the facility may utilize a variety of approaches to applying the patch, including those described in the application incorporated by reference, as well as real-time function call interception, and/or code rewriting of (1) already-loaded executable modules, (2) one or more disk images of the executable module, or (3) instances of executable modules loaded by the operating system's loader. After step 305, the facility continues in step 301 to receive the next patch.

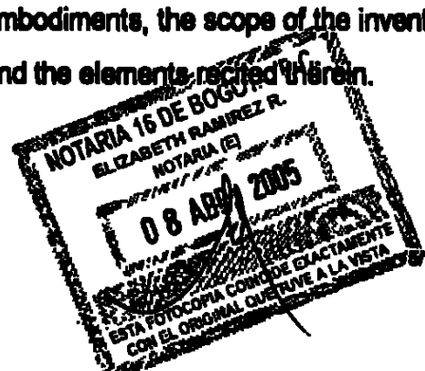
[0052] Figure 5 is a flow diagram showing steps typically performed by the facility in order to update configuration instructions for a particular patch. In step 501, the facility receives configuration instructions for a particular patch, such as from an administrator. In some embodiments, such configuration instructions may be generated by administrators using group policies. In step 502, the facility updates the patch is configuration in the patch table in accordance with the received instructions. After step 502, the facility continues in step 501 to receive the next configuration instructions.

[0053] Figure 6 is a flow diagram showing steps typically performed by the facility to perform the parameter validation specified by a patch. In step 601, a patched function is called. In step 602, if testing is enabled for the patch that affects the called function, then the facility continues in step 603, else the facility continues in step 601 to process the next call to a patched function. In step 603, if test

performance notification is enabled for the patch, then the facility continues in step 604, else the facility continues in step 605. In step 604, the facility generates a notification that the test was performed. Steps 604, 608, and 610 may involve displaying or storing an indication on the target computer system that the test was satisfied, and/or transmitting such an indication to a remote computer system for display or logging there.

[0054] In step 605, the facility performs the validation test specified by the patch. In some embodiments, step 605 involves calling one of a group of standard routines utilized by the facility for testing. In step 606, if the test performed in step 605 is satisfied, then the facility continues in step 601, else the facility continues in step 607. In step 607, if test result notification is enabled for the patch, then the facility continues in step 608, else the facility continues in step 609. In step 608, the facility generates a notification that the test was not satisfied. In step 609, if test result-handling is enabled for the patch, then the facility continues in step 610, else the facility continues in step 601. In step 610, the facility performs the test result handling specified by the patch. After step 610, the facility continues in step 601.

[0055] It will be appreciated by those skilled in the art that the above-described facility may be straightforwardly adapted or extended in various ways. For example, the facility may be used to apply a variety of different kinds of patches in a variety of ways at a variety of locations in executable modules of a variety of types for a variety of purposes. Also, while patches are described herein as containing value validation tests that indicate a problem when they fail, the facility may also be implemented using value invalidity tests that indicate a problem when they succeed. In some embodiments, each test is accompanied by an indication of whether its success or failure indicates a problem. While the foregoing description makes reference to preferred embodiments, the scope of the invention is defined solely by the claims that follow and the elements recited therein.



CLAIM SET 1

We claim:

[c1] 1. A method in a computing system for augmenting software, comprising:

receiving in a target computer system an augmentation specification specifying (a) a function to be augmented, the specified function being among software executed in the computing system, (b) a parameter of the function to be tested, (c) a test to apply to the specified parameter, and (d) a modification to perform to the behavior of the function if the specified test is not satisfied by the specified parameter; and

when the specified function is invoked on the target computer system, if the specified test is not satisfied by the specified parameter, performing the specified modification to the behavior of the specified function.

[c2] 2. The method of claim 1 wherein the modification specified by the augmentation specification is preventing the execution of the specified function.

[c3] 3. The method of claim 1 modification specified by the augmentation specification is executing the specified function after alteration of the specified parameter.

[c4] 4. The method of claim 1, wherein a plurality of augmentation specifications are received in the target computer system.

[c5] 5. The method of claim 4, further comprising maintaining all of the received augmentation specifications in a data structure.

[c6] 6. The method of claim 1 wherein no user intervention is required subsequent to receiving the augmentation specification in order to perform the specified modification to the behavior of the specified function.

[c7] 7. The method of claim 1 wherein the augmentation specification specifies a test to applied to the specified parameter by identifying a parameter testing function to invoke whose code is not included in the augmentation specification.

[c8] 8. The method of claim 1 wherein the augmentation specification specifies a test to be applied to the specified parameter by identifying a parameter testing function to invoke that was installed before the augmentation specification was received.

[c9] 9. The method of claim 1 wherein the augmentation specification contains code.

[c10] 10. The method of claim 9 wherein the code contained by the augmentation specification includes code invoking a parameter testing function whose code is not included in the augmentation specification.

[c11] 11. The method of claim 1 wherein the augmentation specification contains text identifying a parameter testing function whose code is not included in the augmentation specification.

[c12] 12. The method of claim 11 wherein the code of the parameter testing function identified by the text contained by the augmentation specification is contained by the augmentation specification.

[c13] 13. The method of claim 11 wherein the code of the parameter testing function identified by the text not contained by the augmentation specification is not contained by the augmentation specification.

[14. The method of claim 1 further comprising providing an interface for configuring a state for controlling the operation of the augmentation specification, and wherein the specified modification to the behavior of the specified function is performed only when the state is an enabled state.

[c15] 15. The method of claim 14 wherein the provided interface enables the operation of the augmentation specification to be configured locally.

[c16] 16. The method of claim 14 wherein the provided interface enables the operation of the augmentation specification to be configured remotely.

[c17] 17. The method of claim 14 wherein the provided interface enables the operation of the augmentation specification to be configured in accordance with a policy.

[c18] 18. The method of claim 14 wherein the specified test is performed before any substantive code of the specified function is executed.

[c19] 19. The method of claim 1 further comprising providing an interface for configuring warnings associated with the augmentation specification, and wherein, if the provided interface is used to enable warnings, generating a warning each time the test specified by the augmentation specification fails.

[c20] 20. The method of claim 1, further comprising providing an interface for configuring notifications associated with the augmentation

specification, and wherein, if the provided interface is used to enable notifications, generating a notification each time the test specified by the augmentation specification is performed.

[c21] 21. The method of claim 20, further comprising consolidating the generated notifications into a single consolidated notification.

[c22] 22. The method of claim 1 wherein the received augmentation specification is signed,
the method further comprising verifying the signature of the augmentation specification,
and wherein the specified modification to the behavior of the specified function is performed only if verification of the signature of the augmentation specification was successful.

[c23] 23. The method of claim 22 wherein the specified modification to the behavior of the specified function is performed only if the signer of the signature of the augmentation specification matches a signer of the software containing the specified function.

[c24] 24. The method of claim 1, further comprising performing the specified test in response to detecting that the specified function has been invoked, without altering the code of the specified function.

[c25] 25. The method of claim 1, further comprising altering the code of the specified function to perform the specified test when the specified function is invoked.

[c26] 26. The method of claim 25 wherein the code of the specified function is altered by inserting into the code of the specified function a jump to a separate routine that performs the specified test.

[c27] 27. The method of claim 1 wherein the specified function is provided in a distinguished executable module that is loadable using a loader, the method further comprising registering with the loader to have a routine that performs the specified test called before the specified function is executed in response to each invocation of the specified function.

[c28] 28. A computing system that enables augmentation of software present in the computing system, comprising:

a storage device containing software;

a patch receiver that receives in the computing system a patch specifying (a) a point at which the software is to be augmented, (b) a value associated with the function to be tested at the specified point, (c) a test to apply to the specified value, and (d) a modification to perform to the behavior of the software if the specified test is not satisfied by the specified value; and

a patching agent that injects code into the software at the specified point such that, when execution of the software reaches the specified point on the target computer system, if the specified test is not satisfied by the specified value, the specified modification to the behavior of the software is performed.

[c29] 29. A computer-readable medium whose contents cause a target computing system to perform a method for adding value validation to software available in the target computing system, comprising:

receiving in the computing system an augmentation specification specifying (a) a function to which value validation is to be added, the specified function being among software available in the computing system (b) data to be tested that exists in the target computing system during execution of the function,

(c) a test to apply to the specified data, and (d) a modification to perform to the behavior of the function if the specified test is not satisfied by the specified data; and

when the specified function is invoked on the target computer system, if the specified test is not satisfied by the specified data, performing the specified modification to the behavior of the specified function.

[c30] 30. The computer-readable medium of claim 22 wherein the specified test is a test that is always satisfied, irrespective of the value of the specified data.

[c31] 31. A generated data signal conveying a patch data structure, comprising:

information identifying a function to which value validation is to be added;

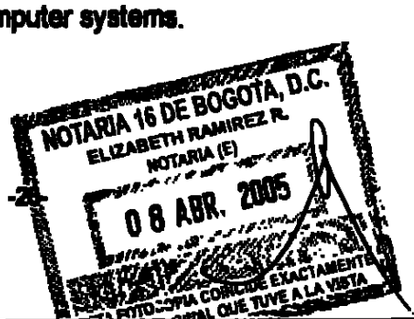
information identifying data to be tested that exists during execution of the function;

information identifying a test to apply to the specified data; and to

information identifying a modification to perform to the behavior of the function if the specified test is not satisfied by the specified data,

such that, if the data signal is received on a target computer system on which the specified function is executed, the contents of the data structure may be used to perform the specified modification to the behavior of the specified function if the specified test is not satisfied by the specified data when the specified function is invoked on the target computer system.

[c32] 32. The generated data signal of claim 31 wherein the generated data signal is transmitted between computer systems.



[c33] 33. The generated data signal of claim 31 wherein the generated data signal is transmitted within a computer system.

[c34] 34. A method for adding value validation to software available in the target computing system, comprising:

receiving in the computing system an augmentation specification specifying (a) software to which value validation is to be added, (b) a point in the specified software at which value validation is to be added, (c) data to be tested that exists in the target computing system during execution of the specified software at the specified point, (d) a test to apply to the specified data, and (e) a modification to perform to the behavior of the software if the specified test is not satisfied by the specified data;

enabling a user to configure an operational mode for the received augmentation specification; and

when the specified software is executed at the specified point on the target computer system, applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification.

[c35] 35. The method of claim 34 wherein the user configures an active operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises, if the specified test is not satisfied by the specified data, performing the specified modification to the behavior of the specified software.

[c36] 36. The method of claim 34 wherein the user configures an active diagnostic operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification

comprises, if the specified test is not satisfied by the specified data, both (1) performing the specified modification to the behavior of the specified software, and (2) generating a notification that the specified test is not satisfied by the specified data.

[c37] 37. The method of claim 34 wherein the user configures a verbose active operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises:

if the specified test is not satisfied by the specified data, performing the specified modification to the behavior of the specified software; and

generating a notification that the specified test was performed.

[c38] 38. The method of claim 34 wherein the user configures an active verbose diagnostic operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises:

if the specified test is not satisfied by the specified data, both (1) performing the specified modification to the behavior of the specified software, and (2) generating a notification that the specified test is not satisfied by the specified data; and

generating a notification that the specified test was performed.

[c39] 39. The method of claim 34 wherein the user configures an inactive operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises omitting to perform the specified modification to the behavior of the

specified software, irrespective of whether the specified test is satisfied by the specified data.

[c40] 40. The method of claim 34 wherein the user configures an inactive diagnostic operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises:

omitting to perform the specified modification to the behavior of the specified software, irrespective of whether the specified test is satisfied by the specified data; and

if the specified test is not satisfied by the specified data, generating a notification that the specified test is not satisfied by the specified data.

[c41] 41. The method of claim 34 wherein the user configures an inactive diagnostic operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises:

omitting to perform the specified modification to the behavior of the specified software, irrespective of whether the specified test is satisfied by the specified data; and

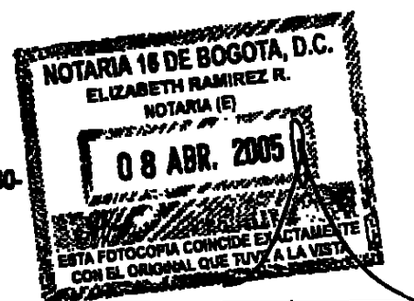
generating a notification that the specified test was performed.

[c42] 42. The method of claim 34 wherein the user configures an inactive diagnostic operational mode for the augmentation specification, and wherein applying the received augmentation specification in a manner consistent with the operational mode configured for the received augmentation specification comprises:

omitting to perform the specified modification to the behavior of the specified software, irrespective of whether the specified test is satisfied by the specified data;

generating a notification that the specified test was performed; and

if the specified test is not satisfied by the specified data, generating a notification that the specified test is not satisfied by the specified data.



EFFICIENT PATCHING

ABSTRACT OF THE DISCLOSURE 1

A facility for augmenting software in a target computer system is described. The facility receives and augmentation specification in the target computer system. The augmentations specification specifies: (a) a function to be augmented, (b) a parameter of the function to be tested, (c) a test to apply to the specified parameter, and (d) and modification to perform to the behavior of the function if the specified test is not satisfied by the specified parameter. When the specified function is invoked on the target computer system, if the specified tested is not satisfied by the specified parameter, the facility performs the specified modification to the behavior of the specified function.

CLAIM SET 2

We claim:

[c43] 1. A method in a computing system, comprising:

receiving in the computing system a distinguished patch package for modifying the behavior of an installed program;

automatically extracting from the distinguished patch package (1) patch application information, identifying a distinguished portion of a distinguished program against which the patch is to be applied, and (2) patch behavior information, specifying a manner in which to modify the behavior of the distinguished portion of the distinguished program; and

automatically adding a distinguished entry to a patch table, the distinguished entry containing the extracting patch application information and patch behavior information.

[c44] 2. The method of claim 1, further comprising using the contents of the distinguished entry to modify the behavior of the distinguished portion of the distinguished program.

[c45] 3. The method of claim 2 wherein the contents of the distinguished entry are used to modify the behavior of the distinguished portion of the distinct program at a time when no administrative user is logged in to the computing system.

[c46] 4. The method of claim 2 wherein the contents of the distinguished entry are used to modify the behavior of the distinguished portion of the distinct program without relying upon any user permissions.

[c47] 5. The method of claim 2 wherein the contents of the distinct entry are used to modify the behavior of the distinguished portion of the distinguished program without determining the location in which the distinguished program is persistently stored.

[c48] 6. The method of claim 2 wherein the contents of the distinguished entry are used to modify the behavior of the distinguished portion of the distinguished program in response to the loading of the distinguished program.

[c49] 7. The method of claim 2 wherein the contents of the distinguished entry are used to modify the behavior of the distinguished portion of the distinguished program in response to the execution of the distinguished portion of the distinguished program.

[c50] 8. The method of claim 2 wherein two instances of the distinguished program are loaded, and the contents of the distinguished entry are used to modify the behavior of the distinguished portion of both instances of the distinguished program.

[c51] 9. The method of claim 1, further comprising:

at the time the patch application information is extracted, determining whether the distinguished program is loaded in the computing system; and

if the distinguished program is loaded in the computing system, using the extracted patch application information to modify the behavior of the loaded distinguished program in accordance with the patch behavior information.

[c52] 10. The method of claim 1, further comprising deleting the distinguished entry from the patch table to prevent the behavior of the distinct

portion of the distinguished program from being modified in accordance with the patch behavior information contained by the distinguished entry.

[c53] 11. The method of claim 1 wherein the extracted patch behavior information specifies modifying the behavior of the distinguished portion of the distinguished program to perform value validation.

[c54] 12. The method of claim 1 wherein the extracted patch behavior information specifies modifying the behavior of the distinguished portion of the distinguished program to perform parameter validation.

[c55] 13. The method of claim 1 wherein the extracted patch behavior information specifies modifying the behavior of the distinct portion of the distinct program to perform validation of a value read from a file.

[c56] 14. The method of claim 1 wherein the extracted patch behavior information specifies modifying the behavior of the distinct portion of the distinct program to perform validation of a value inputted by user.

[c57] 15. The method of claim 1 wherein the extracted patch behavior information specifies modifying the behavior of the distinct portion of the distinct program to perform validation of a value received in one or more network packets.

[c58] 16. The method of claim 1 wherein the extracted patch behavior information specifies modifying the behavior of the distinguished portion of the distinguished program to perform parameter validation by calling a distinguished validation function.

[c59] 17. The method of claim 16 wherein an entry in the patch table other than the distinct entry also contains patch behavior information that

specifies modifying the behavior of the distinct portion of the distinguished program to perform parameter validation by calling the distinguished validation function.

[c80] 18. The method of claim 16, further comprising executing code for the distinguished validation function to modify the behavior of the distinct portion of the distinct program, the distinguished code not being included in the distinguished patch package.

[c81] 19. A computer-readable medium whose contents cause a computing system to perform a method comprising:

receiving in the computing system a distinguished patch package for modifying the behavior of a programmatic entity;

automatically extracting from the distinguished patch package (1) patch application information, identifying a distinguished programmatic entity against which the patch is to be applied, and (2) patch behavior information, specifying a manner in which to modify the behavior of the distinguished programmatic entity; and

automatically adding a distinguished entry to a patch table, the distinguished entry containing the extracting patch application information and patch behavior information.

[c82] 20. The computer-readable medium of claim 19 wherein the distinguished programmatic entity has a plurality of behaviors, only a proper subset of which the patch behavior information specifies modifying, and wherein the received distinguished patch package contains no information about behaviors of the distinguished programmatic entity that the patch behavior information does not specify modifying.

252

[c83] 21. One or more computer memories collectively storing a patch table data structure, comprising a plurality of patch entries, each patch entry containing:

patch application information, identifying a distinguished portion of a distinguished programmatic entity which the patch is to be applied; and

patch behavior information, specifying a manner in which to modify the behavior of the distinguished portion the distinguished programmatic entity,

such that, for a particular patch entry, the contents of the patch entry can be used to modify the behavior of the distinguished portion of the distinguished programmatic entity in the specified manner.

[c84] 22. The computer memories of claim 21 wherein the patch application information identifies an executable module associated with the programmatic entity.

[c85] 23. The computer memories of claim 22 wherein the patch application information identifies a position in the identified executable module.

[c86] 24. The computer memories of claim 23 wherein the patch application information further identifies contents to expect at the identified position in the identified executable module.

[c87] 25. The computer memories of claim 21 wherein a selected patch entry contains a plurality of patch application information instances, each instance identifying a distinguished portion of a different version of the distinguished programmatic entity.

[c88] 26. The computer memories of claim 25 wherein a first one of the patch application information instances identifies a version of the distinguished programmatic entity to which a selected hot fix has been applied, and wherein

second one of the patch application information instances identifies a version of the distinguished programmatic entity to which the selected hot fix has not been applied.

[c69] 27. The method of claim 26 wherein the selected patch entry can be applied to (1) the version of the distinguished programmatic entity to which the selected hot fix has been applied, (2) the version of the distinguished programmatic entity to which the selected hot fix has not been applied, or (3) both, without replacing the version of the distinguished programmatic entity to which the selected hot fix has been applied with the version of the distinguished programmatic entity to which the selected hot fix has not been applied, and without replacing the version of the distinguished programmatic entity to which the selected hot fix has not been applied with the version of the distinguished programmatic entity to which the selected hot fix has been applied.

[c70] 28. The method of claim 24 wherein the selected patch entry may be deployed to a target computer system by an administrator without regard for which version of the distinguished programmatic entity is executed on the target computer system.

[c71] 29. The computer memories of claim 21 wherein the patch behavior information of a selected patch entry contains code usable to modify the behavior of the distinct portion of the distinguished programmatic entity.

[c72] 30. The computer memories of claim 21 wherein the patch behavior information of a selected patch entry contains data usable to modify the behavior of the distinct portion of the distinguished programmatic entity.

[c73] 31. The computer memories of claim 21 wherein the patch behavior information of a selected patch entry specifies a parameter validation

LSA

test to be applied to a specified parameter associated with the distinguished programmatic entity.

[c74] 32. The computer memories of claim 21 wherein the identity of the computer memories storing the patch table data structure is configurable.

[c75] 33. The computer memories of claim 21 wherein the computer memories are directly connected to a computer system on which one or more of the patch entries are to be applied.

[c76] 34. The computer memories of claim 21 wherein the computer memories are directly connected to a computer system on which no patch entries are to be applied.

[c77] 35. A computing system that automatically implements received code patches, comprising:

a pre-installed library of helper functions; and

a patching agent that receives code patches, each code patch targeting a group of executable modules and identifying a helper function in the library, and that, when an executable module in a group targeted by a received code patch is executed, invokes the helper function identified by the code patch targeting the group of executable modules including that executable module to perform value validation.

[c78] 36. The computing system of claim 35 wherein the patching agent includes a library maintenance subsystem that receives new parameter validation functions and automatically adds the received new parameter validation functions to the library, so that these new primary validation functions can be invoked in accordance with code patches that identify them.

[c79] 37. A method in a computing system for automatically applying a software patch, comprising:

receiving in the computing system a patch for modifying the behavior of a programmatic entity, the patch (1) specifying a manner in which to modify the behavior of the programmatic entity, and (2) for each of a plurality of versions of the programmatic entity, (a) identifying the version of the programmatic entity, (b) specifying a position in the version of the programmatic entity to modify the behavior of the programmatic entity in the specified manner, and (c) identifying code expected to reside at the specified position in the version of the programmatic entity before the behavior of the programmatic entity is modified;

automatically determining that a distinguished version of the programmatic entity among the versions of the programmatic entity identified by the patch is installed on the computing system; and

only if the position specified for the distinguished version of the programmatic entity contains the identified code expected to reside at the specified position, modifying the behavior of the programmatic entity at the position specified for the distinguished version of the programmatic entity in accordance with the patch.

[c80] 38. A computer-readable medium whose contents cause a computing system to perform a method for automatically applying a software patch, comprising:

storing in the computing system a patch for modifying the behavior of a programmatic entity, the patch (1) specifying a manner in which to modify the behavior of the programmatic entity, and (2) for each of a plurality of versions of the programmatic entity, (a) identifying the version of the programmatic entity, (b) specifying a position in the version of the programmatic entity to modify the behavior of the programmatic entity in the specified manner, and (c) identifying code expected to reside at the specified position in the version of the programmatic entity before the behavior of the programmatic entity is modified;

automatically determining that a distinguished version of the programmatic entity among the versions of the programmatic entity identified by the patch is installed on the computing system; and

only if the position specified for the distinguished version of the programmatic entity contains the identified code expected to reside at the specified position, modifying the behavior of the programmatic entity at the position specified for the distinguished version of the programmatic entity in accordance with the patch.

EFFICIENT PATCHING

ABSTRACT OF THE DISCLOSURE 2

A facility for automatically processing software patches is described. The facility receives in a computing system a distinguished patch package for modifying the behavior of a programmatic entity. The facility automatically extracts from the distinguished patch package (1) patch application information that identifies a distinguished programmatic entity against which the patches to be applied, and (2) patch behavior information that specifies a manner in which to modify the behavior of the distinguished programmatic entity. The facility automatically adds to a patch table a distinguished entry containing the extracted patch application information and patch behavior information.

CLAIM SET 3

We claim:

[c81] 1. A method in a computing system for applying a software patch, comprising, using an automated patching agent:

receiving the software patch;

in response to receiving the software patch, without user intervention:

identifying a currently-loaded executable module instance to which the received software patch pertains; and

applying the received software patch to the identified currently-loaded executable module instance to modify the behavior of the identified executable module instance.

[c82] 2. The method of claim 1, wherein the identified executable module is an independently-executable module.

[c83] 3. The method of claim 1, wherein the identified executable module is a library.

[c84] 4. The method of claim 1, wherein the identified executable module is a script.

[c85] 5. The method of claim 1, wherein the identified executable module is a managed code module.

[c86] 6. The method of claim 1, wherein two software patches are received pertaining to the same executable module instance, and both received

software patches are applied to modify the behavior of the executable module instance.

[c87] 7. The method of claim 6, wherein the two received patches pertain to different locations in the executable module instance.

[c88] 8. The method of claim 6, wherein the two received patches pertain to the same locations in the executable module instance.

[c89] 9. The method of claim 8, wherein the received patches are applied such that both behavior modifications specified by the patches are exhibited by the patched executable module instance.

[c90] 10. The method of claim 8, wherein the received patches are applied such that only the behavior modification of a dominant one of the two received software patches is exhibited by the executable module instance.

[c91] 11. The method of claim 10, wherein the dominant received software patch contains an explicit indication that it is to dominate one or more other software patches.

[c92] 12. The method of claim 1, wherein the identified executable module instance exhibits the modified behavior at a time after the received software patch is applied.

[c93] 13. The method of claim 1, further comprising, using the automated patching agent, applying the received software patch to a disk image for the identified executable module instance.

[c94] 14. The method of claim 1, further comprising, using the automated patching agent, applying the received software patch to the identified executable module instance when the identified executable module instance is reloaded at a future time.

[c95] 15. The method of claim 1 wherein the currently-loaded executable module instance to which the software patch pertains is identified from among a plurality of executable module instances to which the software patch pertains, and wherein the received software patch is applied using module instance-specific information for the identified executable module instance contained in the received patch.

[c96] 16. The method of claim 15 wherein the plurality of executable module instances are each a different version of the same executable module.

[c97] 17. The method of claim 15 wherein the module instance-specific information for the identified executable module contained in the received patch specifies an offset within the identified executable module at which to modify the behavior of the identified executable module.

[c98] 18. The method of claim 17 wherein the module instance-specific information for the identified executable module contained in the received patch further specifies content of the identified executable module instance that is expected to appear at a specified offset before the received software patch is applied.

[c99] 19. The method of claim 1, further comprising, using the automated patching agent:

receiving a second software patch pertaining to an executable module not then present in the computing system; and

in response to receiving the second software patch, without user intervention, storing the received software patch for future application if an executable module instance to which the received software patch pertains is later loaded in the computing system.

[c100] 20. The method of claim 1, when, after the applying, the identified executable module is uninstalled, and subsequently reinstalled, the method further comprising, using the automated patching agent, without user intervention, reapplying the received software patch to the reinstalled identified executable module instance to modify the behavior of the reinstalled identified execute will module instance.

[c101] 21. The method of claim 1 wherein the received software patch contains code specifying how to modify the behavior of the identified executable module instance.

[c102] 22. The method of claim 1 wherein the received software patch contains data specifying how to modify the behavior of the identified executable module instance.

[c103] 23. The method of claim 1 wherein the received software patch specifies modifying the behavior of the identified executable module instance to add a test and test results-handling logic to a designated portion of the identified executable module instance.

[c104] 24. The method of claim 1 wherein the received software patch specifies modifying the behavior of the identified executable module instance to add parameter validation functionality to a designated function within the identified executable module instance.

[c110] 30. The computer-readable medium of claim 29, wherein the computer-readable medium is a data storage medium.

[c111] 31. The computer-readable medium of claim 29, wherein the computer-readable medium is a data transmission medium.

[c112] 32. A computing system for applying a software patch, comprising:

a receiver that automatically receives the software patch;

an identification subsystem that, in response to received by the receiver of the software patch, without user intervention, identifies a currently-loaded executable module instance to which the received software patch pertains; and

a patch application subsystem that, in response to identification of the currently-loaded executable module instance to which the received software patch pertains by the identification subsystem, without user intervention, applies the received software patch to the identified currently-loaded executable module instance to modify the behavior of the identified executable module instance.

[c113] 33. A computer-readable medium containing a software patch data structure representing a software patch, the software patch data structure comprising:

first information identifying one or more executable software modules to which the patch is to be applied; and

second information specifying a manner in which the identified executable software modules are to be modified,

such that the contents of the software patch data structure may be used by an automatic patching agent to select an executable software module identified by the first information and modify the selected executable software module in the manner specified by the second information.

[c114] 34. The computer-readable medium of claim 33 wherein the computer-readable medium is a data storage medium.

[c115] 35. The computer-readable medium of claim 33 wherein the computer-readable medium is a data transmission medium.

[c116] 36. The computer-readable medium of claim 33 wherein the second information is code specifying a manner in which the identified executable software modules are to be modified.

[c117] 37. The computer-readable medium of claim 33 wherein the second information is data describing a manner in which the identified executable software modules are to be modified.

[c118] 38. The computer-readable medium of claim 33 wherein the software patch data structure further comprises a signature demonstrating that the software patch data structure was produced by a designated patch authority, and that it has not been modified since the signature was generated.

[c119] 39. The computer-readable medium of claim 38 wherein a signer identity can be discerned from the signature of the software patch data structure, and the discerned signer identity can be compared to a signer identity that can be discerned from a signature of the identified executable software module.

EFFICIENT PATCHING

ABSTRACT OF THE DISCLOSURE 3

A facility for applying a software patch is described. Using an automatic patching agent, the facility receives the software patch. In response to receiving the software patch, without user intervention, the facility performs the following acts: First, the facility identifies an instance of an executable module that is currently loaded, and to which the received software patch pertains. Second, the facility applies the received software patch to the identified loaded executable module instance to modify the behavior of the identified executable module instance.

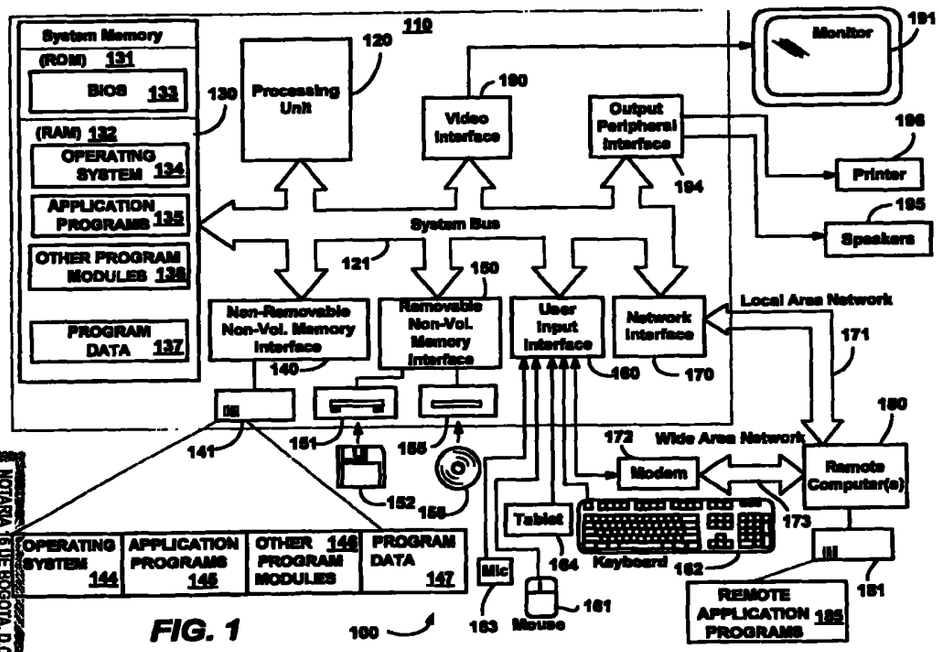


FIG. 1

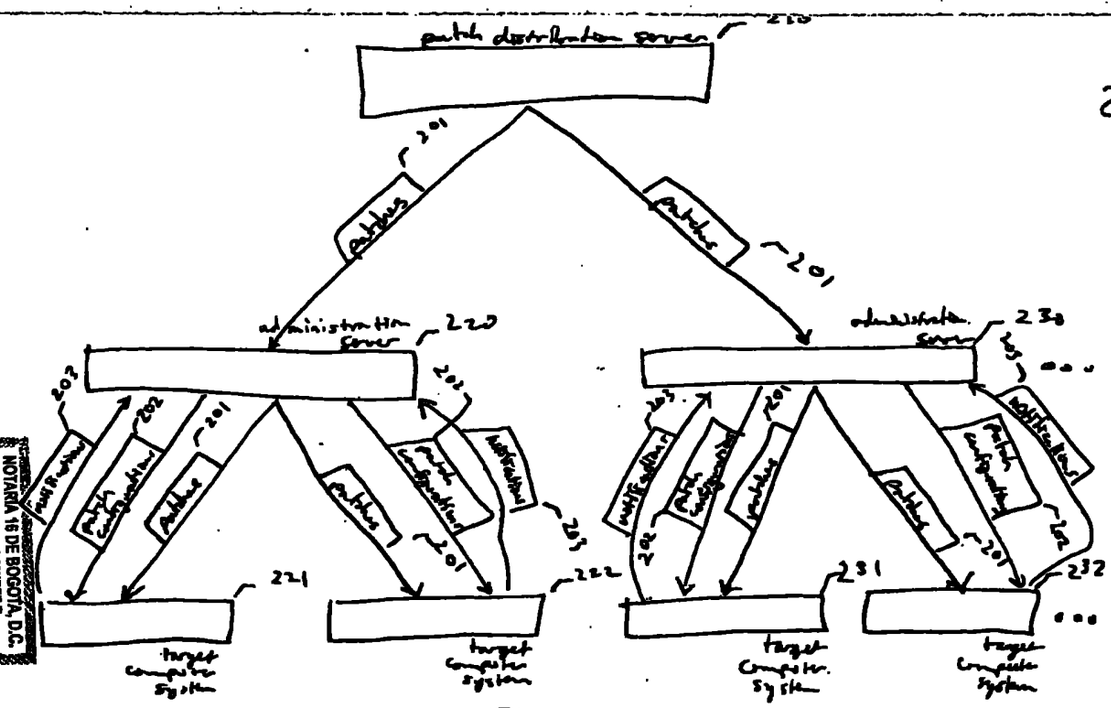


FIG 2

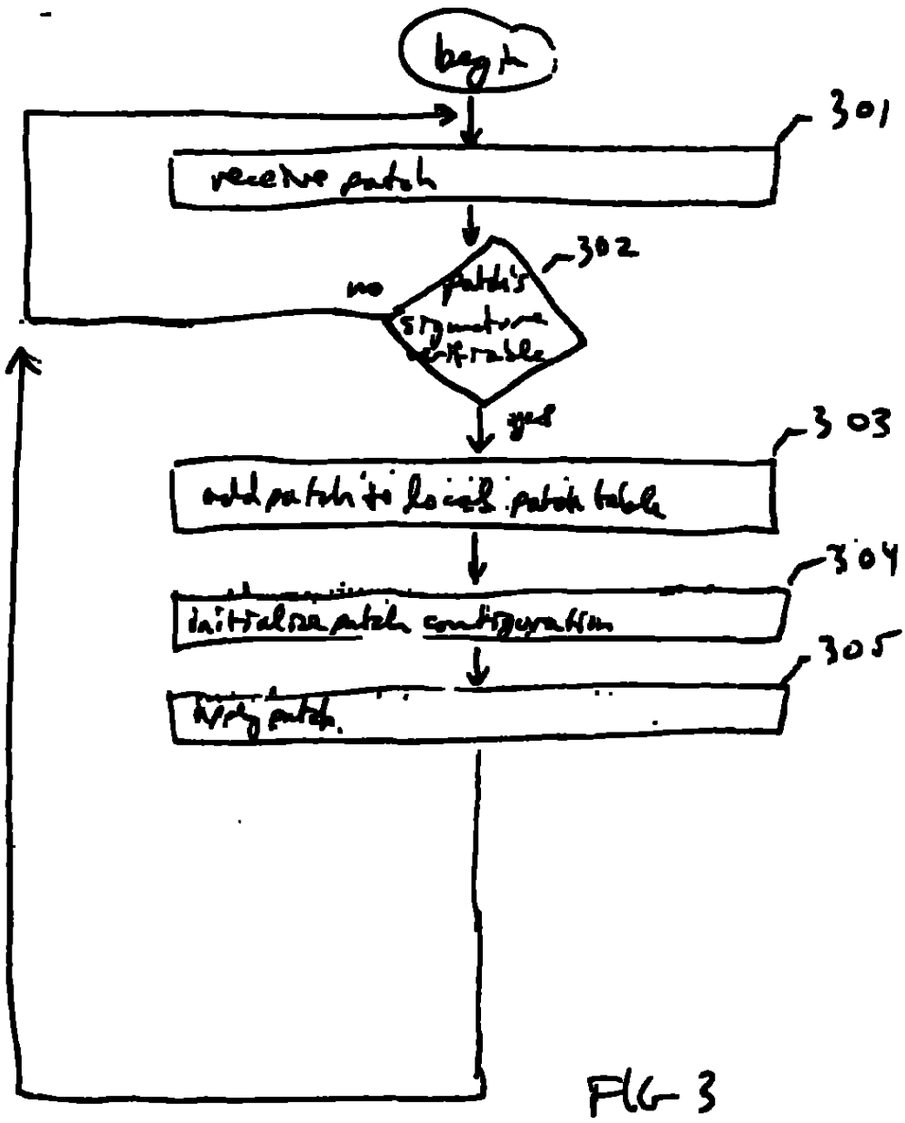


FIG 3

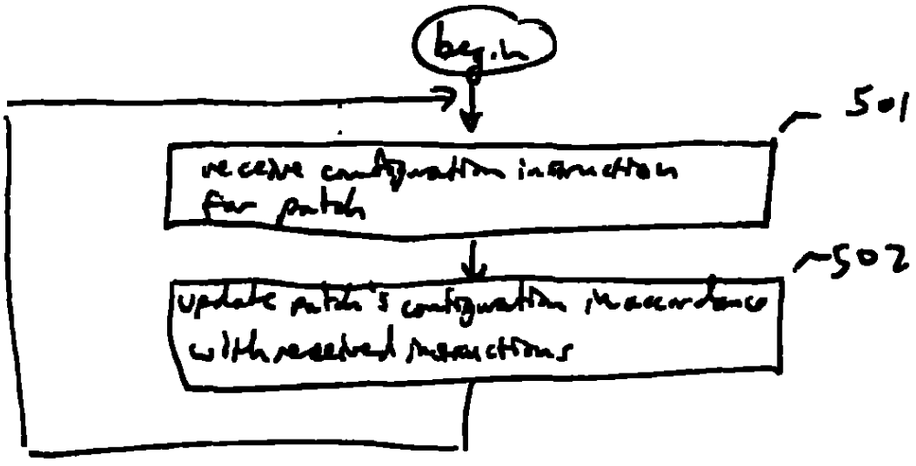


FIG 5

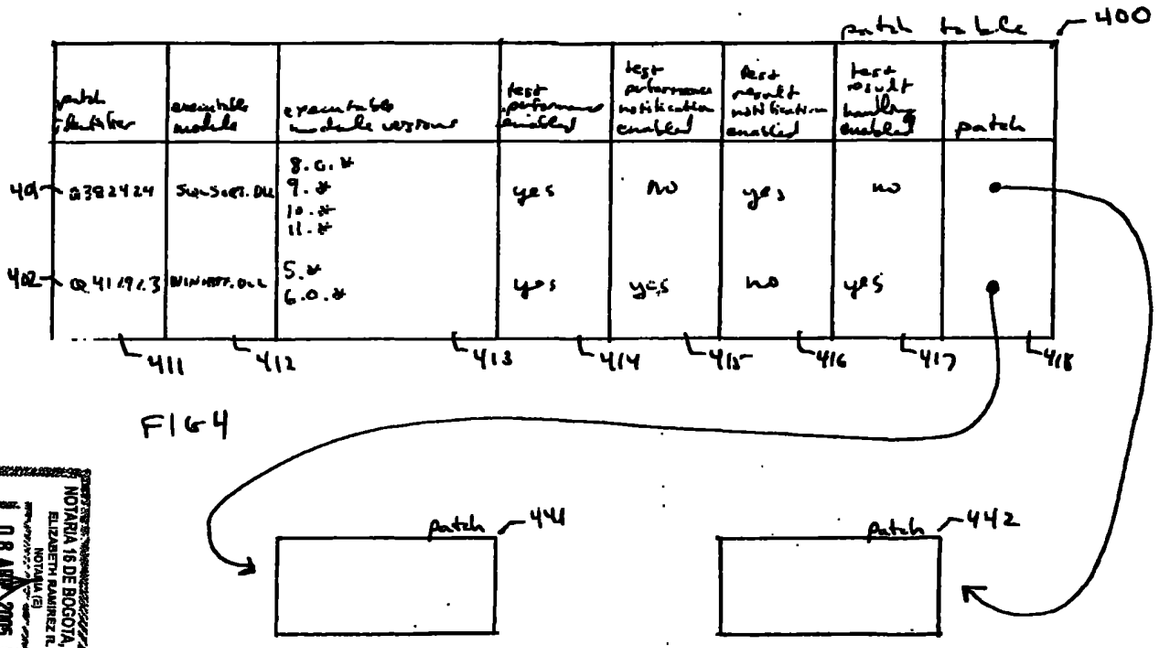
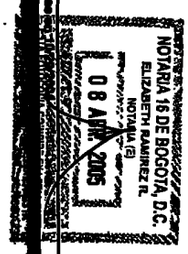


FIG 4



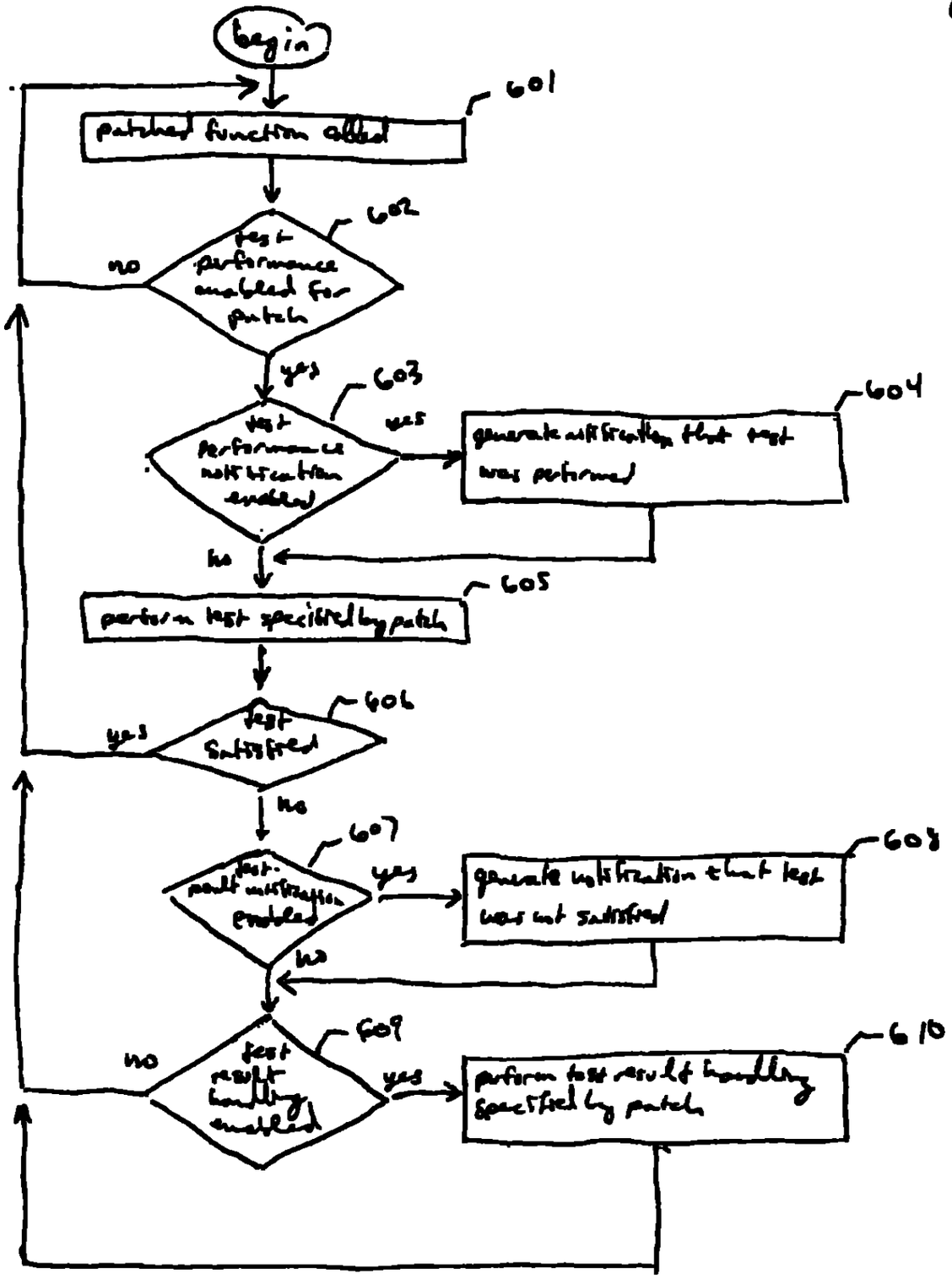


FIG 6

271

REQUISITOS NECESARIOS PARA PRESENTACION Y AGILIZACION DEL TRAMITE

	USUARIO	RADICADOR
PATENTE DE INVENCIÓN MODELO DE ÚTILIDAD	( )	( )
- Indicación de que se solicita la concesión de una patente.	( )	( )
- Datos de identificación del solicitante o de la persona que se presenta la solicitud	( )	( )
- Descripción de la invención.	( )	( )
- Dibujos de ser estos pertinentes.	( )	( )
- Comprobante de Pago de las tasas establecidas.		
COMPLETA ( ) INCOMPLETA	( )	( )
DISEÑO INDUSTRIAL ( )		
- Indicación de que se solicita el registro de Diseño Industrial	( )	( )
- Datos de identificación del solicitante o de la persona que presenta la solicitud.	( )	( )
- Representación gráfica o fotográfica del Diseño Industrial o muestra del Material que incorpora el diseño.	( )	( )
- Comprobante de pago de las tasas establecidas	( )	( )
COMPLETA ( ) INCOMPLETA	( )	( )
ESQUEMA DE TRAZADO ( )		
- Indicación de que solicita el registro de un Esquema de trazado.	( )	( )
- Datos de identificación del solicitante o de la persona que presenta La solicitud.	( )	( )
- Representación gráfica del esquema de trazado	( )	( )
- Comprobante de pago de las tasas establecidas.	( )	( )
COMPLETA ( ) INCOMPLETA	( )	( )

Firma Responsable:

Fecha:

SUPERINDUSTRIA Y COMERCIO
Radicacion : 05A2005 00000000 Folios: 270
Fecha (AMD): 2005-04-11 16:10:14
Tramite : 002 PATENTES 0 1 REGISTRO/ 411 PRESENTA
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

Industria y Comercio
SUPERINTENDENCIA

PROTOCOLO

Expediente 05 - 32905

SUPERINDUSTRIA Y COMERCIO
SISTEMA DE REGISTRO

MANTENIMIENTO DE PROTOCOLOS

Numero : 16069
Fecha : 04/08/1995
Conferido: MICROSOFT CORPORATION

Pais : US ESTADOS UNIDOS DE AMERICA

Ciudad : 234 REDMOND, WASHINGTON

Represent: S Sustituir : S Desistir : S
No. Radicacion : 95 34920 Clase: 0 Seev: 0 Seac: 0

scncia	Codigo	Ctr	Nombre
1	77	A	CARDENAS NAVAS DARIO
2	3653		CARDENAS CABALLERO EDUARDO
3	5048		CARDENAS MARTINEZ BERNARDO

Encontrados (1/1) registro(s)

Carrera 13 No. 27-00 Pisos 5, 7 y 10
Conmutador: 382 0840
Fax: 350 5220
E-mail: info@sic.gov.co
www.sic.gov.co
Bogotá, D.C., Colombia

Industria y Comercio
SUPERINTENDENCIA

Folio 273

2020
Bogotá, D. C.

Doctor(a)
DARIO CARDENAS NAVAS
Apoderado(a)

Oficio No.
012692

REFERENCIA:

Radicación	05 - 32905
Trámite	002
Evento	001
Actuación	430
Folios	001

Como resultado del examen de forma, realizado con fundamento en el artículo 38 de la Decisión 486 de la Comisión de la Comunidad Andina, se le comunica que:

- Allegar copia del documento en que conste la cesión del derecho a la patente otorgado por los inventores al solicitante o a su causante. (art. 26-k,Dec 486/2000).

En cumplimiento del artículo 39 de la Decisión 486, se le requiere para que dentro del término de dos (2) meses siguientes a la fecha de notificación del presente oficio, complete los requisitos anteriormente enunciados. En caso de no dar cumplimiento al presente requerimiento, la solicitud se considerará abandonada y perderá su prelación.

Notifíquese el presente oficio conforme a lo dispuesto en el numeral 5.2, literal e), del capítulo quinto del título primero de la Circular Unica No.10 de 2001.

ALIX CESPEDES DE VERGEL
Jefe de la División de Nuevas Creaciones

El anterior requerimiento fue notificado por fijación en lista de fecha 05 MAYO 2005

202027

Carrera 13 No. 27-00 Pisos 5, 7 y 10
Conmutador: 382 0840
Fax: 350 5220
E-mail: info@sic.gov.co
www.sic.gov.co
Bogotá, D.C., Colombia