

ENVOLTURA CONDICIONAL PARA OBJETO DE PROGRAMA

REFERENCIA CRUZADA A SOLICITUD RELACIONADA

[0001] La presente solicitud reclama prioridad de la solicitud de patente no provisional de Estados Unidos con el número de serie 14/843,908 presentada el 2 de septiembre de 2015 y la solicitud de patente provisional de Estados Unidos con el número de serie 62/046,128 presentada el 4 de septiembre de 2014, la totalidad de los cuales se incorporan aquí como referencia.

ANTECEDENTES

[0002] El desarrollo de un programa de computador es a veces una tarea muy grande y engorrosa. Para cualquier proyecto relativamente grande, hay una cantidad significativa de codificación que debe hacerse, seguida de pruebas y eliminación de errores. Esto típicamente lleva a cambios frecuentes, nuevas mejoras y así sucesivamente, lo cual puede resolver muchos errores pero también puede introducir muchos errores.

[0003] Por ejemplo, considere un programa escrito en JavaScript® o un lenguaje de programación similares de tipos poco rígidos. Los lenguajes de programación que tienen tipos poco rígidos tienen variables que se declaran sin un tipo; el lenguaje de programación determina el tipo para cada variable declarada y no se realiza un chequeo interno de tipos. Además, los tipos son dinámicos, por ejemplo una variable x puede ser asignada a un valor numérico (var x = 10, por lo que el tipo de la variable x es un número) y luego, posiblemente en la siguiente declaración, la misma variable x puede asignarse a una cadena de caracteres (var x = "Martes", por lo cual la

variable x es ahora una cadena de caracteres). Estas características de JavaScript® puede llevar a un número de errores. Por ejemplo, una función existente puede esperar una cadena de caracteres como un parámetro cuando es invocada por una nueva función de llamado, pero esa nueva función de llamado puede por el contrario proporcionar un número, causando un error.

[0004] El código escrito en lenguajes dinámicos, tal como el código de JavaScript®, puede ser más difícil de mantener y depurar de errores debido a su comportamiento dinámico de corrida. Los parámetros de las funciones, los valores de retorno, y los accesorios de propiedad no son impuestos por el tipo, lo que resulta en una gran cantidad de código que requiere manipular una gran cantidad de revisiones dentro de una función para asegurar que los parámetros son del tipo y/o la forma correctos. Adicionalmente, los resultados de funciones son típicamente revisados antes de usarlas debido a esta misma naturaleza dinámica. Para hacer estas revisiones, un código JavaScript® robusto casi siempre se llena de revisiones “if” para validar los datos que entran y los que salen. Sin estas revisiones, es fácil utilizar mal el código y hacer cosas inesperadas (algo puede no fallar inmediatamente, lo cual lleva a problemas subsiguientes). Estas revisiones saturan el código e impactan negativamente el desempeño. El rastreo es otra táctica de prueba/remoción de errores que impacta negativamente el desempeño; rastrear globalmente un programa también lleva un registro de una cantidad considerable de datos, muchos de los cuales no son necesarios cuando se está buscando un problema.

RESUMEN

[0005] Este Resumen se proporciona para introducir una selección de conceptos representativos en una forma simplificada que se describen adicionalmente más abajo en la Descripción Detallada. Este resumen no tiene la intención de identificar

características claves o características esenciales de la materia reclamada, así como tampoco tiene el propósito de ser empleado de alguna manera que pueda limitar el alcance de la materia reclamada.

[0006] En breve, la tecnología acá descrita está dirigida hacia una envoltura condicional, acoplada a un creador de objetos, que envuelve una parte de un objeto (por ejemplo, una función) con lógica adicional. La envoltura condicional es agregada selectivamente al objeto por el creador del objeto al momento de la creación del objeto con base en datos condicionales. Durante el tiempo de corrida, alguna de la lógica agregada puede correr antes que la parte del objeto sea ejecutada y/o alguna después que la parte del objeto ha sido ejecutada. La lógica agregada puede realizar validación y/o rastreo, lo cual puede incluir correr operaciones de validación y/o rastreo antes que la parte del objeto sea ejecutada así como también después que la parte del objeto sea ejecutada.

[0007] Otras ventajas pueden hacerse aparentes de la siguiente descripción detallada cuando se toma junto con los dibujos.

BREVE DESCRIPCION DE LOS DIBUJOS

[0008] La tecnología acá descrita es ilustrada a modo de ejemplo y es limitada por las figuras acompañantes en las cuales los números de referencia indican elementos similares, y en las cuales:

[0009] FIG. 1 es un diagrama de bloques que representa componentes a modo de ejemplo que pueden ser utilizados para proporcionar envolturas condicionales para objetos de programas seleccionados, de acuerdo con uno o más ejemplos de implementaciones.

[0010] FIG. 2 es un diagrama de flujo que muestra ejemplos de pasos que pueden tomarse para envolver un objeto con una envoltura lógica de validación y/o una envoltura lógica de rastreo, o retornar una implementación de objeto sin envoltura, de acuerdo con uno o más ejemplos de implementaciones.

5 [0011] FIG. 3 es un diagrama de bloques que representa ejemplos de conceptos de una envoltura de validación, de acuerdo con uno o más ejemplos de implementaciones.

[0012] FIG. 4 es diagrama de bloques/flujo de datos que representa diferentes aspectos relacionados con el uso de una envoltura de validación durante un tiempo
10 de corrida de una función envuelta, de acuerdo con uno o más ejemplos de implementaciones.

[0013] FIG. 5 es un diagrama de flujo que muestra ejemplos de pasos que pueden tomarse para validar una función mediante una envoltura lógica de validación, de acuerdo con uno o más ejemplos de implementaciones.

15 [0014] FIG. 6 es un diagrama de bloques que representa ejemplos de conceptos de una envoltura de rastreo, de acuerdo con uno o más ejemplos de implementaciones.

[0015] FIG. 7 es un diagrama de bloques/flujo de datos que representa diferentes aspectos relacionados con el uso de una envoltura lógica de rastreo durante el tiempo de corrida de una función envuelta, de acuerdo con uno o más ejemplos de
20 implementaciones.

[0016] FIG. 8 es un diagrama de flujo que muestra ejemplos de pasos que pueden tomarse para rastrear datos y/o una función mediante una envoltura lógica de rastreo, de acuerdo con uno o más ejemplos de implementación.

[0017] FIG. 9 es un ejemplo de un diagrama de bloques de un sistema que puede implementar un patrón de envoltura condicional, de acuerdo con uno o más ejemplos de implementaciones.

[0018] FIG. 10 es un diagrama de bloques que representa un ejemplo de un ambiente de computación dentro del cual pueden incorporarse aspectos de la materia descrita aquí.

DESCRIPCIÓN DETALLADA

[0019] Diferentes aspectos de la tecnología acá descrita están dirigidos generalmente hacia una envoltura condicional de objeto que envuelve una parte de un objeto (por ejemplo, el objeto mismo (por ejemplo, su nombre y tipo de información), una función dentro de él, un evento y así sucesivamente) con código que activa una acción relacionada con una evaluación respecto a la parte envuelta del objeto, cuando se ejecuta durante el tiempo de corrida. Por ejemplo, la envoltura condicional puede comprender una envoltura de validación que ocasiona que datos relacionados a una función, incluyendo parámetro(s) de entrada y/o valor(es) de retorno de salida, sean validados durante el tiempo de corrida, atrapando así cualquier error ocasionado por datos inapropiados. Como otro ejemplo, la envoltura condicional puede comprender una envoltura de rastreo que ocasiona que se registren puntos de rastreo (por ejemplo, sean archivados) antes, durante y/o después que una parte del objeto sea ejecutada durante el tiempo de corrida.

[0020] La envoltura condicional puede aplicarse selectivamente solo a ciertos objetos o ciertas partes de objetos, por ejemplo, en el momento de la creación del objeto con base a unos datos condicionales. Por ejemplo, aunque la envoltura condicional puede aplicarse globalmente a los objetos de un programa como un

todo, los datos de la condición pueden especificar que la envoltura condicional solo se puede aplicar a objetos en un área determinada del programa, por ejemplo, una que está dirigida a una sub-tarea. La envoltura condicional puede agregarse a partes de un objeto, por ejemplo, la información del objeto, una o más funciones, propiedades, eventos, y así sucesivamente; (note que no solo pueden envolverse tipos típicos de funciones, sino que propiedades que tienen funciones de obtener y fijar también pueden envolverse para evaluación por validación y/o rastreo como otros métodos)

[0021] En una o más implementaciones, un patrón de envoltura corresponde a un

Lugar central (por ejemplo, un sistema que crea objetos de JavaScript® a partir de definiciones de tipo de clase) que crea los objetos de un programa. Por ejemplo, el patrón de envoltura puede volver la llamada de una función en una forma tal que emite una nueva función que contiene lógica adicional, y aún así sigue siendo capaz de llamar a la función original. Éste patrón de envoltura por lo tanto permite que las funciones sean envueltas selectivamente/Condionalmente, tal como para correr una construcción de eliminación de errores de un programa (Validar y/o rastrear código mientras está corriendo) en comparación con una construcción de producción (si validación o rastreo). El patrón envoltura puede aplicarse al año y un programa tal como uno basado en un espacio de nombre/clase mi nombre, a un nombre método/propiedad (una notación de variable privada), y demás. Algo de tal información pueden mantenerse en un archivo separado u otra estructura de datos asociada con el objeto. Por ejemplo, un probador puede especificar que todas las funciones dentro de un aria del programa que realizan alguna tarea del programa (por ejemplo, manejar la comunicación de la red) sea validada; las funciones/declaraciones de validación a utilizar son identificadas en cada uno de los archivos de los documentos de la función, pero el probador sólo necesita suministrar información al OOP del sistema una vez (Por ejemplo, identificar un nombre de

espacio de un objeto que identifica los objetos que manejan la comunicación de la red) con el fin de encender la validación para estas funciones.

[0022] cuando se vuelve parte de un objeto, lógica adicional causa la validación, rastreo, y/o activar virtualmente cualquier otra cosa que un desarrollador/probador/eliminador de errores pueda considerar útil. Por ejemplo, la lógica adicional puede poner el programa en pausa y notificar al desarrollador/probador. Cuando no se envuelve, la función original se crea y sin modificación, permitiendo el máximo desempeño cuando se utiliza durante el tiempo de corrida. Al utilizar el patrón de envoltura en la creación de un objeto y sus partes (por ejemplo, funciones, propiedades, y funcionalidad extendida de sintaxis/marco específico de un lenguaje), hay un punto central para la validación extendida y el rastreo a todo lo largo del programa de la aplicación de una manera uniforme.

[0023] respecto a la validación, los tipos de definiciones puede usar el patrón de envoltura para seleccionar propiedades, método, y demás. El patrón de envoltura decir si utiliza o no la versión de implementación envuelta o la implementación original con base en virtualmente cualquier dato de condición especificada por un desarrollador/probador. Por ejemplo, las condiciones específicas pueden ser evaluadas cuando se decide si se debe envolver, incluyendo el tener un archivo de desarrollador o prueba que especifique cuales funciones y/o propiedades deben ser validadas y/o rastreadas, cuál área de un programa valide, cuales objetos serán modelados durante la prueba (por lo que la validación o el rastreo de objetos modelados es generalmente innecesaria) y así sucesivamente. Pueden considerarse las condiciones externas, por ejemplo, habilitar la validación para una función que a veces falla durante condiciones de poca memoria, o cuando hay una conexión lenta a la red, y así sucesivamente, para ayudar a depurar esa función.

[0024] Con respecto a la validación, la validación puede extenderse más allá del chequeo del tipo de variable y utilizarse para asegurar que los datos de entrada/datos de salida son de otro modo válidos. Un ejemplo es para revisar que un valor de datos de entrada/datos de salida está dentro de un cierto rango (por ejemplo) entre 0 y 100), es valido con respecto a una numeración (por ejemplo, es un valor identificado dentro de una enumeración de códigos de estado http) y así sucesivamente. Otro ejemplo es una validación de tipos que verifica si las entradas/salidas son creadas a partir de un sistema específico (por ejemplo, uno que procesa definiciones de clases dentro de objetos de JavaScript®) y que el caso tiene un nombre de espacio/nombre (o tipo derivado) específico en ese sistema. Adicionalmente, puede haber una validación que impone que un caso implemente una interfaz específica. Como se entiende, la envoltura condicional puede usarse para validar virtualmente cualquier cosa que pueda ser verificada.

[0025] Una vez la validación es efectuada exitosamente en un programa, no es necesario que la validación sea usada en ese programa, por lo que el código resultante es más limpio, ya que no necesita verificaciones de validación allí que atiborren el código. Más aún, la validación que se hace en los datos de entrada y los datos de salida es más consistente que efectuar verificaciones en el código tradicional. Para facilitar la consistencia, en una o más implementaciones la validación de una parte de un objeto viene de la documentación (por ejemplo, un objeto de Notación de Objeto de JavaScript® (JSON) que describe el objeto), tal que las funciones de validación y/o las declaraciones relacionadas con la validación son recogidas automáticamente durante la operación de envoltura. Con la validación mediante documentación, es más difícil utilizar mal el código de JavaScript ® y obtener resultados inesperados de las llamadas de API.

[0026] Con respecto al rastreo, muchos lenguajes no proporcionan un buen mecanismo de rastreo , lo cual hace más difícil la depuración en ciertas plataformas

(tal como dispositivos sin una solución de depurado remoto sencilla). Posibilitar el rastreo a través del tablero es costoso, ya que todo llamado de una función tiene algún impacto de desempeño negativo asociado con el rastreo, y se genera una gran cantidad de datos.

5 [0027] Al utilizar un patrón de envoltura, hay un lugar central para determinar si y cuando rastrear llamadas de funciones basado en datos de condición. Una función envuelta puede rastrear información asociada con la llamada de la función antes de llamar la implementación de la función original, y nuevamente rastrear la información que la llama. Esto proporciona puntos de rastreo que pueden usarse para depurar
10 el comportamiento de un programa de aplicación, y llevar el tiempo de la información para suministrar más contexto sobre el desempeño del código.

[0028] Debido al sobre costo asociado con el rastreo de la información, la funcionalidad de la envoltura puede decidir si y cuando realizar el rastreo, por ejemplo, siempre que un método es llamado o cambia el valor de una propiedad,
15 por ejemplo. Como con la validación, se pueden emplear criterios de condición para activar selectivamente el rastreo, tal como de qué área del código viene una función. En una o más implementaciones, un archivo de configuración de entradas (por ejemplo parte de los datos de condición) puede ser accedido para determinar si el creador del objeto necesita generar el código de rastreo adicional para cada método
20 o propiedad.

[0029] Considere, por ejemplo, un desarrollador que trata de encontrar un problema asociado con la entrada de un teclado en un programa de aplicación. El rastreo es habilitado selectivamente para las clases relacionadas con la entrada, pero deshabilitado para otras. Esto permite que se procese una cantidad más pequeña
25 de información de rastreo (y enviada remotamente de ser necesario), lo cual no

impacta el desempeño en la misma manera en que lo haría la habilitación del rastreo de manera global.

[0030] Con respecto a las propiedades, determinar cuando y donde se establecen las propiedades puede ser una tarea tediosa, sin embargo hacerlo es a veces significativo cuando se encuentran problemas. Al posibilitar el rastreo sobre un fijador de propiedades específico, es mucho menos difícil ver cuando y donde se fija una propiedad. Esto es similarmente cierto para un recuperador de propiedades.

[0031] Debe entenderse que cualquiera de los ejemplos aquí no es limitante. Por ejemplo, muchos de los ejemplos se refieren a validar el(los) parámetro(s) de entrada y el(los) resultado(s) de salida de una función (método) durante la validación, y/o validar los valores de una propiedad, así como también rastrear funciones y fijadores de propiedades. Sin embargo, virtualmente cualquier entidad describible en el código puede ser validada, incluyendo pero sin limitarse a funciones, recuperadores de propiedades, fijadores de propiedades, valores de propiedades, eventos de interfaz, valores de entrada y salida, tipos de valores de entrada y salida, forma de los datos (por ejemplo, formato) y demás. Más aún, cualquier código que se ejecuta puede rastrearse. Como tal, la tecnología descrita acá no está limitada a las modalidades, aspectos, conceptos, estructuras, funcionalidades o ejemplos particulares descritos aquí. Por el contrario, cualquiera de las modalidades, aspectos, conceptos, estructuras, funcionalidades o ejemplos descritos acá no son limitantes, y la tecnología puede usarse de diferentes maneras que proporcionan beneficios y ventajas en la computación y la programación en general.

[0032] La FIG. 1 es un diagrama de bloques que muestra un creador de objetos que crea objetos basado en archivos de documento, que representan clases, subclases y así sucesivamente. Como se describe acá, basado en uno o más datos

de condición 106 (por ejemplo, fijados por un desarrollador o un código de prueba), varias o diferentes partes de cualquier objeto de un programa de aplicación 108 pueden envolverse con código relacionado con la validación y/o código relacionado con el rastreo (y/o posiblemente otro código relacionado con evaluación del objeto);
5 estos se muestran en la FIG. 1 como los objetos envueltos 110. Para objetos en los cuales los datos de condición no se cumplen, se crean los objetos 112 para el programa de aplicación en su forma original (sin envolver).

[0033] Como se muestra en la FIG. 1, puede envolverse cualquier parte de un objeto. Un objeto puede envolverse con código de validación que valida que el
10 objeto es parte de cierto marco/tipo de sistema, y código de validación para las otras partes. Un objeto puede ser envuelto con código relacionado con el rastreo. Por ejemplo, el objeto A de los objetos envueltos 110 representados incluye una fijación de propiedad 114 que esta envuelta con una envoltura 116 y una función 118 que está envuelta con una envoltura 120. Note que es factible tener una parte de un
15 objeto envuelta con tanto una envoltura de validación como una envoltura de rastreo; típicamente si se usan ambas, la validación esta por fuera de la envoltura de rastreo tal que el rastreo no rastrea las operaciones relacionadas con la validación, (a menos que se deseara rastrear las operaciones relacionadas con la validación en adición con la parte del objeto).

20 [0034] Como se muestra en la FIG. 1, el envolver no es necesariamente se lleva a cabo en toda parte de un objeto envuelto; por ejemplo, la función 122 del objeto A no está envuelta. En general, el código de desarrollador/prueba puede selectivamente encender la envoltura para cualquier parte del objeto o conjunto de partes del objeto (por ejemplo, para envolver todas las funciones en el área del
25 programa que se encarga de las comunicaciones HTTP).

[0035] También como se muestra en la FIG. 1, solo unos selectos de los objetos pueden ser envueltos. Otros objetos 122 y la función 124 del objeto B se muestran como son, sin envoltura, y en el ejemplo de la FIG. 1 ninguna otra parte del objeto B o cualquiera de los objetos 112 están envueltas. Tales objetos sin envolver se
5 corren, por lo tanto, de manera normal sin el sobre costo resultante de ejecutar el código de envoltura y sus operaciones relacionadas de validación y/o rastreo, por ejemplo.

[0036] La FIG. 2 muestra lógica en forma de ejemplos de pasos de un diagrama de flujo que pueden emplearse durante la creación del objeto para determinar si hay
10 envoltura o no para un objeto que está siendo creado, comenzando con el paso 202 donde se recibe una solicitud para crear un objeto y el documento o documentos que describen el objeto son accedidos para crear el objeto y sus diferentes partes, por ejemplo, propiedades, funciones, eventos y así sucesivamente. Note que cada clase o similares utilizados para crear el objeto típicamente tienen su propio
15 documento, sin embargo, un objeto puede derivar de otro objeto/clase, y por lo tanto puede necesitarse más de un documento para determinar la información relacionada con la validación y/o el rastreo para el objeto.

[0037] EL paso 204 representa evaluar una o más condiciones como se ejemplifican aquí las cuales determinan si alguna parte del objeto debe ser envuelta. Si no, el
20 paso 204 se ramifica al paso 206 en donde el objeto es devuelto a su forma original (sin envoltura) a la entidad que lo pide, por ejemplo, el programa de aplicación al momento del lanzamiento o durante el tiempo de corrida.

[0038] Si al menos alguna parte del objeto debe ser envuelta, incluyendo la información general del objeto (por ejemplo, validar si esta pertenece
25 apropiadamente a un sistema de tipo basado en su nombre completamente cualificado, el cual es identificado en una parte del objeto y está registrado con el

sistema de tipo), el paso 208 selecciona la primera parte del objeto. El paso 210 evalúa si los datos de condición indican que esta parte debe envolverse con código de envoltura de rastreado. De ser así, el paso 212 envuelve esta parte con código de envoltura de rastreado.

5 [0039] El paso 214 evalúa si los datos de condición indican que la parte seleccionada debe envolverse en código de envoltura de validación. De ser así, el paso 216 envuelve la parte con código de envoltura de validación. Note que si se utilizan ambas envolturas, la validación generalmente no hace parte del rastreo, por ejemplo, las operaciones previas a la validación (antes de correr la entidad
10 rastreada) y las operaciones posteriores a la validación (después de correr la entidad rastreada) no son rastreadas son en sí mismas, a menos que se desee, en cuyo caso se hace de manera directa.

[0040] Los pasos 218 y 220 repiten las operaciones de envoltura para cada parte del objeto. De esta manera, cualquier propiedad, método, evento, constructor y
15 demás pueden ser envueltos. Cuando el objeto es envuelto como lo especifican los datos de condición, el paso 222 regresa el objeto envuelto a la entidad que lo solicita. Debe notarse que los pasos 204 y 206 son opcionales por eficiencia, en el hecho que un objeto que no es envuelto nunca tendrá los pasos 212 y/o 216 ejecutados para ninguna parte del mismo, y en esta situación el paso 222 devolverá
20 en su lugar un objeto sin envolver. Sin embargo, si el creador del objeto sabe que la envoltura del objeto no está especificada para el objeto, no es necesario pasar cada parte separadamente.

[0041] La FIG. 3 ejemplifica la validación de la envoltura, y muestra una clase A 332 de ejemplo que va a convertirse a un objeto (por ejemplo, un objeto de JavaScript®)
25 por ejemplo, por un sistema de creación de objeto 334 al momento de creación del objeto. Aunque el JavaScript® es prototípico y no tiene clases, un sistema OOP,

descrito aquí con referencia a la FIG. 9, puede ejecutar tal operación. Como antes, el envolver un objeto dado o cualquier parte del mismo con una envoltura de validación depende de los datos de condición que permiten la envoltura de validación 336.

5 [0042] En el ejemplo de la FIG. 3, cualquiera del método 338 o recuperador de propiedad 340 o fijador de propiedad 342 puede ser selectivamente envuelto con una envoltura de validación 344. Los eventos, datos de propiedad, constructores y demás también pueden ser envueltos. Si se habilita para una función tal como la de método 338, por ejemplo, la lógica de envoltura de validación 344 devuelve una
10 función modificada que 1) comprueba los argumentos contra documentación para la validación (tipo, requerimiento, forma, etc.), 2) corre la implementación de la función original y 3) comprueba el resultado de la función original contra la documentación (tipo, requerimiento, forma, etc.). Operaciones similares de validación pueden llevarse a cabo con el fijador y recuperador de propiedad 340 y
15 342 ejemplificados, (así como también para otras entidades de objetos tales como eventos). Si no se habilita, la lógica de envoltura de validación 344 no modifica la implementación del objeto original.

[0043] El objeto 346 devuelto es por lo tanto la implementación original o está envuelto de alguna manera con código de validación. Note que la FIG. 3 es
20 solamente respecto a la envoltura de validación; el objeto 346 devuelto puede no tener envoltura de validación, pero puede estar envuelto con envoltura de rastreo, como se describe aquí.

[0044] Las FIGS. 4 y 5 muestra como generalmente opera la validación durante el tiempo de corrida, utilizando el ejemplo de la función 440 envuelta con una envoltura
25 de validación 442 como parte de un objeto 444. Para cualquier función, típicamente hay dos fases de validación, a saber una validación previa que chequea si el los

parámetros de entrada se adhieren a los requerimientos de entrada de la función, y la validación posterior que chequea si el o los valores devueltos cumplen los requerimientos de salida de la función. Note que la función (ordinariamente) no se le permite correr a menos que pase la validación previa; (aunque es factible permitirle a una función proceder, tal como cuando se desea probar como maneja una función malos datos de entrada).

[0045] Por lo tanto, cuando una función envuelta 440 (FIG. 4) es invocada con uno o más parámetros de entrada (paso 502 de la FIG. 5), se realiza la validación de la entrada, como lo representa la flecha marcada (1) en la FIG. 4. En una o más implementaciones, esto involucra acceder (directa o indirectamente) a la información previa a la validación 450 en un documento de archivo (objeto JSON) 452 asociado con el objeto envuelto 444 que indica cómo debe validarse la función. La información de validación previa 450 y la información de validación posterior 458 (por ejemplo, funciones de validación, declaraciones y/o valores de datos para comparación) pueden ser copiados directamente en la lógica de la envoltura de validación 442 al momento de creación del objeto, esto es, los bloques 450 y 458 de la FIG. 4 pueden estar contenidos dentro del bloque 442, o la lógica de la envoltura de validación (o el creador del objeto) sabe cómo acceder a esta información cuando lo necesita. Como se puede apreciar, cualquier entidad puede ser dispuesta para participar en cualquier nivel, así que por ejemplo el código de la envoltura de validación puede pedir que la validación se efectúe externamente y obtener de vuelta un resultado, o puede participar en la validación interna hasta al menos alguna extensión en la corrida/invocación del código de validación).

[0046] En cualquier evento, la información de validación previa 450, puede especificar, por ejemplo, que una función de validación o conjunto de funciones de validación sea invocado (como los representa el paso 504 de la FIG. 5). La función puede ser suministrada mediante una librería de función de validación 456. En lugar

de o adicionalmente a una función de validación o conjunto de funciones de validación, la información de la validación previa 450 alternativamente puede incluir un conjunto de declaraciones que son ejecutadas para realizar la validación. Por ejemplo, la información de validación previa 450 puede incluir una declaración que
5 pasa la validación si el resultado de alguna función de validación A ejecutó un AND donde el resultado de alguna función de validación B sea cierto.

[0047] En este ejemplo, la validación se lleva a cabo (interna o externa a la envoltura de validación 442, o alguna combinación de tanto interna como externa) con un estado de resultado de Cierto (Pasa) o Falso (Falla) (Flecha 2A)). En la situación
10 donde la validación previa falla, un error 460 es la salida (flecha (2B) en la FIG. 4). Esto también se representa en el ejemplo del diagrama de flujo de la FIG. 5 mediante los pasos 506 y 508.

[0048] En la situación donde la validación previa pasa, la función original es corrida por la envoltura (flecha (3) en la FIG. 4, paso 510 and la FIG. 5). En el paso 512 de
15 la FIG. 5, se obtiene(n) el(los) valor(es) de retorno de la función/resultado(s). La validación posterior, la cual también puede ser una función de validación invocada conjunto de funciones de validación y/o declaraciones de validación) luego revisan el resultado de la función en los pasos 514 y 516. Esto también se muestra en la FIG. 4 mediante (las flechas (4) y (5A)).

[0049] En la situación donde la validación posterior falla, la salida es un error 462 (flecha (5B) en FIG. 4). Esto también está representado en ejemplo del diagrama de flujo de la FIG. 5 con los pasos 516 y 518.

[0050] En la situación donde pasa la validación previa, se devuelve(n) el(los) valor(es)/resultado(s) de la función. Esto se representa mediante la flecha (6) en la
25 FIG. 4 y en el ejemplo del diagrama de flujo de la FIG. 5 mediante los pasos 516 y 520.

[0051] A modo de ejemplo de una revisión de validación, un valor variable puede ser revisado para verificar que es el tipo correcto y tiene un valor correcto. Por ejemplo, un parámetro de entrada que requiere ser un código de estado HTTP válido puede ser revisado durante la validación previa para verificar que es del tipo número y tiene un valor que cumple una numeración que comprende los códigos de estado HTTP válidos. Un valor de retorno puede requerir ser un número entre 0 y 100; la revisión de tipo para un número junto con el que tenga un valor que cae dentro del rango especificado es por lo tanto llevada a cabo durante la validación previa. Como se puede apreciar con facilidad, otras partes de objetos tales como fijadores y recuperadores de propiedad, eventos, constructores y demás pueden ser también revisados para el tipo, revisados para el valor y así sucesivamente para la validación (aunque el concepto de validación previa y validación posterior puede no aplicar, tal como si se revisara para ver si un valor de propiedad es o no una cadena antes que se ejecute cualquier parte del objeto).

[0052] Debe notarse que el sistema de validación descrito acá permite validaciones complejas, por ejemplo, utilizando operadores Booleanos tales como AND y OR. Por ejemplo, una función puede realizar una revisión de tipo, con su resultado unido mediante el operador AND con el resultado de otra función que determina si un valor corresponde con una numeración (o cae dentro de un rango y así sucesivamente – básicamente cualquier cosa que especifique un desarrollador); solo si ambos son Verdaderos se devuelve el resultado Verdadero en este ejemplo. Un fijador de propiedad puede, por ejemplo, fijar una propiedad como un número de valor o usar el operador OR para fijarla a un valor de cadena (pero no, por ejemplo, un tipo Booleano, un tipo nulo, un tipo indefinido o un tipo símbolo); la validación, por lo tanto, debe comprobar que el valor que se va a fijar es un valor de número o, mediante el operador Booleano OR, un valor de cadena, con el Verdadero devuelto si una (o ambas) son Verdaderas.

[0053] La FIG. 6 ejemplifica envoltura de rastreado, y muestra un ejemplo clase Z 662 que debe convertirse a un objeto (por ejemplo, un objeto JavaScript®), por ejemplo, por un sistema de creación de objeto 664 en el momento de la creación del objeto. Como sucede antes, el deber o no envolver un objeto dado cualquier parte del mismo con una envoltura de validación depende de los datos de condición que permiten la envoltura de validación 666.

[0054] En el ejemplo de la FIG. 6, cualquier método 668 o (recuperador de 670 o fijador de 342) propiedad puede envolverse selectivamente en una envoltura de rastreado 674. Los eventos, constructores y similares también pueden ser envueltos. Si se posibilita para una función tal como el método 668, por ejemplo, la envoltura de rastreo 674 devuelve una función modificada que 1) inicia un rastreado y/o registra datos de rastreado indicando el nombre de la función con información sobre los argumentos, 2) corre la implementación de la función original, y 3) detiene el rastreado y/o registra los datos del rastreado después de indicar el resultado del nombre de la función con información sobre el (los) valor(es) devueltos. Operaciones similares de rastreado pueden efectuarse con el fijador y el recuperador de propiedad 670 y 672, (así como también para otras entidades de objetos tales como eventos). Si no se habilita, la envoltura de rastreo 674 no modifica la implementación del objeto original (aunque puede estar envuelto en una validación y no terminar de hecho como el original; por brevedad, no se usa validación en este ejemplo). En el ejemplo de la FIG. 6 el objeto devuelto 676 es por lo tanto la implementación original o envuelto de alguna manera con código relacionado con el rastreado.

[0055] La FIG. 7 es un ejemplo de un diagrama de bloques dirigido hacia aspectos del rastreado, utilizando un ejemplo de una función 772 como la parte del objeto a ser rastreada. En la FIG. 7, la función 772 está envuelta con una envoltura de rastreo 774, la cual puede acceder a la información de rastreo 776 (si está disponible, tal

como en la documentación asociada con el objeto) para obtener cualquier información variable indicando constantes/consideraciones de rastreado; por ejemplo, cuales tipos de mensajes registrar, y cuales sacar por filtrado, ya sea si se registra antes o después de los puntos de rastreado o también trazar la ejecución del código.

[0056] dependiendo del propósito del rastreado, el rastreado puede no ser encendido durante la ejecución del código. Por ejemplo, para evaluar el desempeño de una función, puede considerarse más preciso el ingresar el tiempo de inicio de una función y posiblemente el estado de algunos datos (por ejemplo, los valores del parámetro de entrada) antes de correr la función, y registrar el tiempo final de la función y posiblemente el estado de algunos datos (por ejemplo, los valores de retorno de la función); puede que no se realice ningún rastreado durante la corrida de la función, ya que el rastreado puede dar falsos datos de desempeño. Alternativamente, se pueden almacenar el tiempo de inicio/datos de inicio como puntos de inicio del rastreado, mientras que el rastreado se encuentra encendido durante la ejecución de la función para capturar diferentes mensajes de rastreado, junto con almacenar el tiempo final/datos finales como un punto final del rastreado. Cómo realizar el rastreado puede ser mantenido en la información de rastreado 776 para un objeto, por ejemplo, en los archivos de los documentos asociados con el objeto. Puede que incluso no haya ninguna información variable diferente a la de rastrear una función o similares, en cuyo evento por defecto el inicio del rastreado es registrado con datos relacionados con el inicio, el rastreado se enciende (o alternativamente no se enciende por defecto), se corre la función, se apaga el rastreado, y el final del rastreado se ingresa junto con datos relacionados con el final.

[0057] Como se muestra en la FIG. 7, mediante la envoltura de rastreado 774 (lógica) la función envuelta 772 puede trazar información asociada con el llamado de la función en un registro 708 o similar antes de invocar la implementación de la

función original (flecha (1)), y después de invocar la función original (flecha (3)). La información de rastreo puede ser recogida durante la ejecución de la función (flecha (2)). La envoltura 774 proporciona así puntos de rastreo que pueden utilizarse para depurar el comportamiento de cualquier parte de una aplicación, y

5 llevar el tiempo de la información para darle algo de contexto sobre como está el desempeño del código. La envoltura 774 también es capaz de encender y apagar el rastreo respecto a trazar la ejecución de la función. En cualquier evento, el rastreo condicional como se describe acá limita la cantidad de datos que resultan del rastreo, por ejemplo, solo a los datos de puntos de rastreo registrados antes

10 y después de correr la función, junto con los datos de rastreo capturados mientras se corría la función (pero no capturados para la totalidad de la ejecución del código).

[0058] La FIG. 8 muestra un diagrama de flujo con pasos de ejemplo relacionados con el rastreo, comenzando en el paso 802 donde se ejecuta el código de la envoltura previo al rastreo. Esto puede, por ejemplo, incluir operaciones

15 representadas por el bloque 804, para registrar datos del tiempo de inicio y entrada, y opcionalmente encender el rastreo con el fin de trazar la ejecución de la función.

[0059] El paso 806 corre la implementación de la función original que está envuelta con el código de envoltura. (También es factible correr una función envuelta con código de envoltura de validación, sin embargo esto no se hace en el ejemplo de la

20 FIG. 8)

[0060] El paso 808 ejecuta el código de la envoltura posterior al rastreo. Esto puede, por ejemplo, incluir las operaciones representadas por el bloque 810, por ejemplo, el registrar el tiempo final y los datos de salida de la función, y opcionalmente apagar el rastreo (si está encendido) con el fin de limitar los

25 mensajes de rastreo solo a aquellos resultantes de la envoltura y registrados basándose en la ejecución de la función.

[0061] Volviéndonos a otro aspecto, los proyectos que tienen sus propios marcos de trabajo para simplificar la creación de clases/tipos pueden simplificar su código de validación permitiendo a la envoltura condicional hacer esto por ellos, así como también aplicar selectivamente el rastreado. Al combinar el patrón de la envoltura con otros elementos específicos del marco de trabajo, se debe lograr una solución de validación rica.

[0062] Por ejemplo, en la FOG. 9 un sistema tipo OOP 970 tiene un conjunto de información específica del tipo del marco de trabajo que puede ser utilizada por el patrón de la envoltura. En general el sistema OOP 970 toma plantillas 972 (por ejemplo, correspondiente a definiciones de clase) y produce archivos de código fuente 974, archivos de documentos 976 (por ejemplo, objetos JSON) y archivos de prueba 978. Estos archivos son mantenidos en un sistema de documentos o similares. También puede generarse documentación que es más “legible por humanos”.

[0063] Para producir las diferentes salidas, el sistema OOP 90 incluye varias funciones, incluyendo una función de definición de clase 980, una función constructora 981, una función de propiedades 982, una función de eventos 983, una función de métodos 984 y otras funciones, por ejemplo, funciones 985 que soportan otras características orientadas al objeto tal como numeraciones, banderillas, interfaces y demás. También se suministra el código que maneja la envoltura condicional 988 para adicionar lógica de envoltura condicional al objeto como se describe acá.

[0064] Durante un tiempo de lanzamiento y/o para el desarrollo/(prueba/depurado (por ejemplo no el tiempo de corrida de la construcción de producción donde los objetos son establecidos para cada versión de producto del programa), cuando la petición de la creación de un objeto 990 viene como se describe acá, los datos de

condición 992 se utilizan para determinar si se agrega o no lógica de envoltura condicional 994 al objeto. Como también se describe acá, al menos parte de los datos de condición 992 pueden ser especificados in la carpeta o carpetas del documento 976 asociadas con el objeto que se está requiriendo, así como datos relacionados con la validación y el rastreado, por ejemplo, cuales funciones de validación a realizar, cómo rastrear y así sucesivamente, Alternativamente, dicha información puede mantenerse en un archivo u otra estructura de datos separados asociada con el objeto.

[0065] En general, en la creación de un objeto el sistema OOP 970 / código de manejo de la envoltura 988 utiliza datos de condición 992 para decidir si envuelve o no el objeto, y si se determina envolver, cómo envolver el objeto. El objeto pedido es por lo tanto devuelto como el objeto devuelto 996, ya sea envuelto o sin envolver como se describe acá.

[0066] Note que la validación puede incluir la validación en los tipos e instancias OOP (tipos derivados, interfaces implementadas por esa instancia, etc.) y los eventos OOP con validación en los argumentos cuando se dispara el evento. Un tipo de numeración OOP asegura que el valor suministrado es un valor de numeración válido. La validación aplicada a las funciones y propiedades ha sido ejemplificada también acá.

[0067] Como se puede ver, se ha descrito una tecnología en la cual un patrón de envoltura envuelve parte(s) de objeto(s) seleccionados basado en datos de condición, tal como funciones de objetos, en el momento de la creación del objeto. Cuando se envuelve, la lógica adicional puede agregar validación, rastreado, (y otra lógica que puede ser útil) que es ejecutada durante el tiempo de corrida. Para objetos que no están envueltos, la función original del objeto se utiliza sin modificación durante el tiempo de corrida.

[0068] Uno o más aspectos están dirigidos hacia determinar si se debe o no envolver un objeto con código de envoltura para agregar lógica a la parte del objeto para la ejecución de la lógica durante el tiempo de corrida, incluyendo hacer una determinación de si se debe o no envolver basado en datos de condición de envoltura. Si la determinación es envolver, acá se describe la envoltura de parte del objeto con código de envoltura para retornar una implementación de objeto envuelto. Si la determinación es no envolver, se devuelve una implementación de objeto sin envolver.

[0069] Los datos de condición de envoltura pueden indicar que la parte del objeto debe ser validada durante el tiempo de corrida, por lo cual la determinación es envolver. La envoltura de la parte del objeto con el código de envoltura puede incluir envolver el código del objeto con código de envoltura de validación que comprende lógica relacionada con la validación. Cuando la parte del objeto es una función, la lógica relacionada con la validación puede incluir lógica de validación previa que se ejecuta antes de invocar la función y lógica de validación posterior que se ejecuta después de invocar la función. Por ejemplo, correr el código de envoltura de validación previa puede validar uno o más parámetros de entrada y correr el código de validación posterior puede validar uno o más valores de retorno de la función. Correr el código de envoltura de validación previa resulta en invocar al menos una función de validación.

[0070] Cuando los datos de condición de envoltura indican que la parte del objeto debe tener datos de rastreo mantenidos para la parte del objeto cuando la parte del objeto es ejecutada durante el tiempo de corrida, la determinación es envolver. Envolver la parte del objeto con el código de envoltura puede incluir envolver el código del objeto con código de envoltura de rastreo. El código de envoltura puede incluir iniciar el rastreo de la ejecución de la parte del objeto, ejecutar la parte del objeto, y detener el rastreo. El código de envoltura puede estar dirigido

hacia registrar la información de rastreo que representa un punto de inicio del rastreo relacionado con la parte del objeto y registrar otra información de rastreo que representa un punto de parada del rastreo relacionado con la parte del objeto. El registro de la información que representa el punto de inicio del rastreo puede incluir registrar datos relacionados con el inicio, y registrar la información que representa un punto de parada del rastreo puede incluir el registro de un tiempo de parada y el registro de datos relacionados con la parada.

[0071] La parte del objeto puede ser una función, registrar la información de rastreo que representa el punto inicial del rastreo relacionado con la función puede incluir registrar datos de parámetros de entrada a la función, y registrar la información de rastreo que representa un punto de parada de rastreo relacionado con la parte de objeto puede incluir registrar uno o más valores de retorno de la función.

[0072] Los datos de condición de envoltura pueden indicar que la parte del objeto debe ser validada y rastreada durante el tiempo de corrida, por lo que la determinación es envolver. La envoltura de la parte del objeto con el código de envoltura puede incluir envolver el código del objeto con lógica de envoltura de rastreo y envolver el código del objeto con lógica de envoltura de validación.

[0073] Uno o más aspectos están dirigidos hacia un creador de objetos que crea un objeto y una envoltura condicional, acoplados al creador del objeto, que envuelve una parte del objeto con lógica adicional. La envoltura condicional es agregada selectivamente al objeto por el creador del objeto al momento de la creación del objeto basado en los datos condicionales. La envoltura condicional está configurada para correr la lógica agregada antes que la parte del objeto sea ejecutada durante el tiempo de corrida y/o después que la parte del objeto es ejecutada durante el tiempo de corrida.

[0074] La lógica agregada puede ser configurada para disparar una validación de los datos a ser usados por la parte del objeto antes que la parte del objeto sea ejecutada, y/o puede ser configurada para activar una validación de los datos devueltos por la parte del objeto después que la parte del objeto ha sido ejecutada.

5 [0075] La lógica agregada puede ser configurada para activar un salvado de datos relacionados con el rastreo con respecto a la parte del objeto antes que la parte del objeto sea ejecutada. La lógica agregada puede ser configurada para activar un guardado de datos relacionados con el rastreo generados durante la ejecución de parte del objeto.

10 [0076] La lógica agregada puede ser configurada para activar una validación de los datos correspondientes a la parte del objeto y para activar un guardado de datos relacionados con el rastreo que corresponden a la parte del objeto.

[0077] Uno o más aspectos están dirigidos hacia envolver una función con código de validación previa y código de validación posterior, en el cual la función es
15 seleccionada para ser envuelta en base a datos condicionales asociados con la función al momento de la creación del objeto. Durante el tiempo de corrida, el código de validación previa determina si se corre o no la función. Si al correr el código de validación previa se determina que la función no debe correr, la salida es un error. Si al correr el código de validación previa se determina que se debe correr la función,
20 acá se describe que se corre la función, y se corre código de validación posterior para determinar si se devuelve o no un resultado de la función. Si al correr el código de validación posterior se determina que el resultado de la función no debe retornarse, la salida es un error; si al correr el código de validación posterior se determina que el resultado de la función se debe retornar, se retorna el resultado de
25 la función.

[0078] La función puede ser envuelta con código de rastreo basado en datos de condición. Envolver la función con el código de rastreo puede incluir agregar lógica que guarda datos del punto de inicio del rastreo antes de correr la función y que salva datos del punto de parada del rastreo después de correr la función.

5

DISPOSITIVO DE COMPUTO DE EJEMPLO

[0079] Las técnicas descritas acá pueden aplicarse a cualquier dispositivo o conjunto de dispositivos (máquinas) capaces de correr programas y procesos. Se puede entender, por lo tanto, que los computadores personales, portátiles, de mano, portables y otros dispositivos de cómputo y objetos de computo de todo tipo incluyendo teléfonos celulares computadores de tableta/tabla, consolas de juegos/entretenimiento y similares están contempladas para uso en conexión con diferentes implementaciones incluidas aquellas ejemplificadas acá. En concordancia, el mecanismo de computo de propósito general descrito debajo en la FIG. 10 es solo un ejemplo de un dispositivo de computo.

[0080] Las implementaciones pueden ser implementadas parcialmente mediante un sistema de operación, para su uso por un desarrollador de servicios para un dispositivo u objeto, y/o incluidas dentro de software de aplicaciones que opera para realizar uno o más aspectos funcionales de las diferentes implementaciones descritas acá. El software puede describirse en el contexto general de instrucciones ejecutables por computador, tal como estaciones de trabajo de clientes, servidores y otros dispositivos. Aquellos con habilidad en el arte apreciarán que los sistemas de cómputo tienen una variedad de configuraciones y protocolos que pueden ser empleados para comunicar datos, y por lo tanto ninguna configuración o protocolo se considera limitante.

[0081] la FIG. 10 por lo tanto ilustra un ejemplo de un ambiente de sistema de cómputo 1000, en el cual uno o aspectos de la implementación descrita acá puede ser implementada, aunque como se deja en claro arriba, el ambiente del sistema de cómputo 1000 es solo un ejemplo de un ambiente de computo adecuado y no tiene la intención de sugerir ninguna limitación al alcance o uso de la funcionalidad. En adición, el ambiente del sistema de cómputo 1000 no tiene la intención de ser interpretado como que tiene alguna dependencia relacionada con uno cualquiera o combinación de los componentes ilustrados en el ambiente del sistema de cómputo 1000 del ejemplo.

[0082] Con referencia a la FIG. 10, un ejemplo de un dispositivo para implementar una o más implementaciones incluye un dispositivo de cómputo de propósito general en forma de un computador 1010, Los componentes del computador 1010 pueden incluir, pero no se limitan a, una unidad de procesamiento 1020, una memoria de sistema 1030 y un sistema de USB 1022 que acopla varios componentes del sistema a la unidad de procesamiento 1020.

[0083] El computador 1010 típicamente incluye una variedad de medios legibles por máquina (por ejemplo, un computador y puede ser cualquier medio disponible que puede ser accesado por una máquina tal como un computador 1010. La memoria del sistema 1030 puede incluir medios de almacenamiento de computador en forma de memoria volátil y/o memoria no volátil tal como memoria de solo lectura (ROM) y/o memoria de acceso aleatorio (RAM), y medios de disco duro, medios de almacenamiento óptico, medios flash, y así sucesivamente; como se utiliza acá los medios de almacenamiento legibles por máquina / legibles por computador almacenan los datos que no incluyen las señales transitorias, (aunque otros tipos de medios legibles por máquina / legibles por computador que nos memoria pueden hacerlo). A modo de ejemplo, y no limitante, la memoria del sistema 1300 puede

también incluir un sistema operativo, programas de aplicaciones, otros módulos de programas, y datos de programas.

[0084] Un usuario puede ingresar comandos e información en el computador 1010 mediante uno o más dispositivos de entrada 1040. Un monitor u otro tipo de dispositivo de pantalla también está conectado al bus del sistema 1022 mediante una interface, tal como la interface de salida 1050. En adición a un monitor, los computadores también pueden incluir otros dispositivos periféricos de salida tal como parlantes y una impresora, los cuales pueden estar conectados mediante la interfaz de salida 1050.

[0085] El computador 1010 puede operar en una red o un ambiente distribuido utilizando conexiones lógicas a uno o más de otros computadores remotos, tal como el computador 1070. El computado remoto 1070 puede ser un computador personal, un servidor, un enrutador, un computador de red, un dispositivo par u otro nodo de red común, o cualquier otro dispositivo de consumo o transmisión de medios, y puede también incluir cualquiera o todos los elementos descritos arriba en relación con el computador 1010. Las conexiones lógicas mostradas en la fig. 10 incluyen una red 1072, tal como una red de área local (LAN) o una red de área amplia (WAN), pero también puede incluir otras redes/buses. Tales ambientes de red con comunes en hogares, oficinas, redes de computadores corporativas, intranets y el Internet.

[0086] Como se menciona arriba, si bien las implementaciones de los ejemplos han sido descritas en conexión con diferentes dispositivos de cómputo y arquitecturas de red, los conceptos subyacentes pueden ser aplicados a cualquier sistema de red y dispositivo o sistema de cómputo en el que sea deseable implementar esta tecnología.

[0087] También, hay múltiples maneras de implementar una funcionalidad igual o similar, por ejemplo, un API, juego de herramientas, código impulsor, sistema

operativo, control, objeto de software solo o descargable apropiados, etc., los cual le permite a las aplicaciones y servicios tomar ventaja de las técnicas suministradas acá. Por lo tanto, las implementaciones son contempladas desde el punto de vista de un API (u otro objeto de software), así como desde un objeto de software o equipo que implemente uno o más implementaciones como se describen acá. Así, las diferentes implementaciones descritas acá pueden tener aspectos que están completamente en el equipo, parcialmente en el equipo y parcialmente en el software, así como completamente en el software.

[0088] La palabra “ejemplo” se utiliza acá para significar que sirve como un ejemplo, caso o ilustración. Para evitar dudas, la materia divulgada acá no está limitada por tales ejemplos. Además, cualquier aspecto o diseño descrito acá como “ejemplo” no debe necesariamente tomarse como preferido o ventajoso sobre otros aspectos o diseños, y tampoco tiene como propósito excluir estructuras y técnicas ejemplares equivalentes conocidas por aquellos con habilidad en el arte. Más aún, hasta el punto donde los términos “incluye”, “tiene”, “contiene” y otras palabras similares son utilizadas, para evitar dudas, tales términos tienen la intención de incluir de una manera similar al término “que comprende” como una palabra de transición abierta sin excluir ningún elemento adicional o diferente cuando se emplea en las reivindicaciones.

[0089] Como se menciona, las diferentes técnicas descritas acá pueden implementarse en conexión con equipo o software o, cuando sea apropiado, con una combinación de ambos. Como se utiliza acá, los términos “componente”, “módulo”, “sistema” y similares tiene también la intención de referirse a una entidad relacionada con un computador, ya sea equipo, una combinación de equipo y software, o software en ejecución. Por ejemplo, un componente puede ser, pero no se limita a, un proceso que corre en un procesador, un procesador, un objeto, un ejecutable, un hilo de ejecución, un programa, y/o un computador. A modo de

ilustración, tanto una aplicación que corre en un computador como el computador pueden ser un componente. Uno o más componentes pueden residir dentro de un proceso y/o hilo de ejecución y un componente puede estar localizado en un computador y/o distribuido entre dos o más computadores.

5 [0090] Los sistemas antes mencionados han sido descritos con respecto a la interacción entre los diferentes componentes. Puede apreciarse que tales sistemas y componentes pueden incluir aquellos componentes o sub-componentes especificados, algunos de los componentes o sub-componentes, y/o componentes adicionales, y de acuerdo con las diferentes permutaciones y combinaciones de lo
10 anterior. Los sub-componentes también pueden implementarse como componentes acoplados de manera comunicativa a otros componentes en lugar de estar incluidos dentro de componentes parentales (jerárquicos). Adicionalmente, puede notarse que uno o más componentes pueden ser combinados en un solo componente que proporciona una funcionalidad agregada o dividido en varios sub-componentes
15 separados, y que una cualquier o más capas intermedias, tal como una capa administrativa, pueden suministrarse para acoplar de manera comunicativa tales sub-componentes con el fin de proporcionar funcionalidad integrada. Cualquier componente acá descrito puede también interactuar con uno o más componentes no descritos específicamente acá pero generalmente conocidos por aquellos con
20 habilidad en el arte.

[0091] En vista de los ejemplos de sistemas descritos acá, también se pueden apreciar metodologías que pueden ser implementadas con referencia a las tablas de flujo / diagramas de flujo de las diferentes figuras. Aunque por propósitos de simplicidad en la explicación, las metodologías se muestran y se describen como
25 una serie de bloques, debe entenderse y apreciarse que las diferentes implementaciones no están limitadas por el orden de los bloques, ya que algunos bloques pueden ocurrir en diferente orden y/o concurrentemente con otros bloques

a partir de lo que se muestra y describe acá. Donde se ilustra un flujo no secuencial o ramificado mediante las tablas de flujo / diagramas de flujo, puede apreciarse que pueden implementarse otras ramas, rutas de flujo, y orden de los bloques los cuales logran el mismo resultado o uno similar. Más aún, algunos bloques ilustrados son
5 opcionales en la implementación de las metodologías acá descritas.

CONCLUSION

[0092] Aunque la invención es susceptible de diferentes modificaciones y construcciones alternativas, ciertas implementaciones ilustradas de las mismas se
10 muestran en los dibujos y han sido descritas arriba en detalle. Debe sin embargo entenderse, que no hay una intención de limitar la invención a las formas específicas divulgadas, sino por el contrario, la intención es cubrir todas las modificaciones, construcciones alternativas, y equivalentes que caen dentro del espíritu y alcance de la invención.

15 [0093] Adicionalmente a las diferentes implementaciones descritas acá, también debe entenderse que otras implementaciones similares pueden usarse o pueden hacerse modificaciones y adiciones a la(s) implementación(es) descrita(s) para realizar una función igual o equivalente de la(s) implementación(es) correspondiente(s) sin desviarse de la misma. Aún más, múltiples circuitos de
20 procesamiento o múltiples dispositivos pueden compartir el desempeño de una o más funciones descritas acá, y similarmente, el almacenamiento puede efectuarse a través de una pluralidad de dispositivos. En concordancia, la invención no está limitada a una sola implementación cualquiera, sino más bien debe tomarse en amplitud, espíritu y alcance de acuerdo con las reivindicaciones adjuntas.