Industria y Comercio

SUPERINTENDENCIA

IE

SOLICITUD
PATENTE DE INVENCIÓN

E-XPEDIENTE N° 04-128509

TITULO: SOPORTE DE VERSIONES EN LENGUAJES Y HERRAMIENTAS DE PROGRAMACION ORIENTADAS A OBJETO

CLASIFICACION INTERNACIONAL: G06F 17/60

SOLICITANTE: MICROSOFT CORPORATION

DOMICILIO: REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA

APODERADO: DARIO CARDENAS NAVAS

BOGOTÁ, D.C. _____

PETITORIO

Industria y Comercio
SUPERINTENDENCIA

SUPERINDUSTRIA Y COMERCIO
Radicacion : 04126309 00000000 Folios: 118
Fecha (AMD): 2004-12-24 10:26:26
Tramite : 002 PATENTES B 1 REGISTRO/ 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

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	Número
	Lugar de Constitución	Estado de Washington, Estados Unidos de América	
	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.
4 **INVENTOR(ES) (72)**	Nombre:	1. Anthony S. Williams 2. C. Douglas Hodges 3. Clemens A. Szyperski 4. James S. Millar 5. John J. Rivard 6. Jonathan C. Hawkins 7. Patrick H. Dussud 8. Srivatsan Parthasarathy 9. William G. Evans	**IDENTIFICACION** C.C. [ ] NIT [ ]
	Dirección:	1. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America 2. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America 3. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America 4. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America 5. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America 6. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America 7. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America 8. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America 9. One Microsoft Way Redmond, Washington 98052, Estados Unidos de America	
	Teléfono:	3123600 Fax: 3123600	C.E. [ ] Otro [X]
	E-Mail:	general@cardenasycardenas.com	Número
	Domicilio:	1. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA. 2. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA. 3. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA. 4. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA. 5. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA. 6. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA. 7. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA. 8. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA. 9. REDMOND, WASHINGTON, ESTADOS UNIDOS DE AMERICA.	

Título(54):

SOPORTE DE VERSIONES EN LENGUAJES Y HERRAMIENTAS DE PROGRAMACION ORIENTADAS A OBJETO

	(33)País de Origen	(32)Fecha	(31)N° de Solicitud
(6) **Prioridad**	U.S.A.	Febrero 5, 2004	10/772992

SI [X] NO [ ]

(7) **Para publicar a partir de la fecha de la presente solicitud a los:**

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

(8) **Comprobante de Pago No.:** 04-62,210, 04-91,149, 04-98,664 **Fecha** Agt. 4, 2004, Nov. 24, 2004, Dic. 22, 2004

ANEXOS

- [X] Comprobante de pago de la tasa de presentación de la solicitud.
- [ ] 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.
- [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)
- [ ] Copia certificada de la primera solicitud, si se reivindica prioridad.
- [ ] Traducción simple de la primera solicitud, si se reivindica prioridad.
- [ ] 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.

(10)

NOMBRE: DARIO CARDENAS NAVAS

FIRMA:

C.C. 17.068.629 BTA TP. 1.250 C.S.J.

3

SUPERINDUSTRIA Y COMERCIO
Radicacion : 04128589 0000000 Folios: 118
Fecha (AMD): 2004-12-24 10:05:26
Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

NIT : 900174099-2

RECIBO OFICIAL DE CAJA : 04 - 62,210
FECHA : AGOSTO 4 DE 2004

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	32558873	04/08/2004	4,670,000.00

C O N C E P T O

CANT.	RENTISTICO	CONCEPTO	TOTAL CONCEPTO
1	30005-01-01 SOLICITUDES	3 TRAMITES DE SOL. DE MODELO DE UTI	248,000.00
		TOTAL : $	248,000.00

SON: DOSCIENTOS CUARENTA Y OCHO MIL PESOS

RESPONSABLE : _____

RECIBO DE CAJA APLICADO AL EXPEDIENTE No. _____

SOPORTE DE VERSIONES EN LENGUAJES Y HERRAMIENTAS DE PROGRAMACION ORIENTADAS A OBJETO

CARDENAS & CARDENAS

PROCESO

N

4

NIT : 800176089-2 SUPERINDUSTRIA Y COMERCIO

Radicacion : 04128589 0000000 Folios: 118

Fecha (AMD): 2004-12-24 10:26:26

RECIBO OFICIAL DE CAJA : 04 - 91,149

FECHA : NOVIEMBRE 24 DE 2004

Tramite : 000 PATENTES D 1 REGISTRO 411 PRESENTAC

Dependencia: 2000 DIVISION DE NUEVAS CREACIONES

***** 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	28716293	24/11/2004	3,559,000.00

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

CANT.	RENTISTICO	CONCEPTO	TOTAL CONCEPTO
1	50005-01-06 REGISTRO DE CESION, CAMBIO NOM	15 NOMBRES Y ENSENAS COMERCIALES,NAR	213,000.00
		TOTAL : $	213,000.00

SON: DOSCIENTOS TRECE MIL PESOS

RESPONSABLE : _____

RECIBO DE CAJA APLICADO AL EXPEDIENTE No.___ - - - - - - - -

SOPORTE DE VERSIONES EN LENGUAJES Y HERRAMIENTAS DE PROGRAMACION ORIENTADAS A OBJETO

CARDENAS & CARDENAS

ABOGADOS

3

5

SUPERINDUSTRIA Y COMERCIO
Radicacion : 04126509 00000000 Folios: 118
Fecha (AMD): 2004-12-24 18:26:26
Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO

NIT : 800176089-2

CERTIFICACION DE EXCEDENTE : 04 - 98,664
FECHA : DICIEMBRE 22 DE 2004

***** 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	37962499	22/12/2004	1,153,000.00

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

CANT.	RENTISTICO	CONCEPTO		TOTAL CONCEPTO
1	30003-01-09 OTROS SERVICIOS DE PROPIEDAD I	99 OTROS SERVCIOS ADMINISTRATIVOS		79,000.00
		TOTAL :	$	79,000.00

SON: SETENTA Y NUEVE MIL PESOS

RESPONSABLE : _____

CARDENAS & CARDENAS ABOGADOS

RECIBO DE CAJA APLICADO AL EXPEDIENTE No._____

SOPORTE DE VERSIONES EN LENGUAJES Y HERRAMIENTAS DE PROGRAMACION ORIENTADOS A OBJETO

ESTA CERTIFICACION DE EXCEDENTE DE PAGO SOLO PODRA SER UTILIZADA COMO PARTE DE LA CANCELACION DE UNA TARIFA VIGENTE Y NO PODRA SER SUSTITUIDA POR NINGUN RECIBO.

MTC Final

12x12

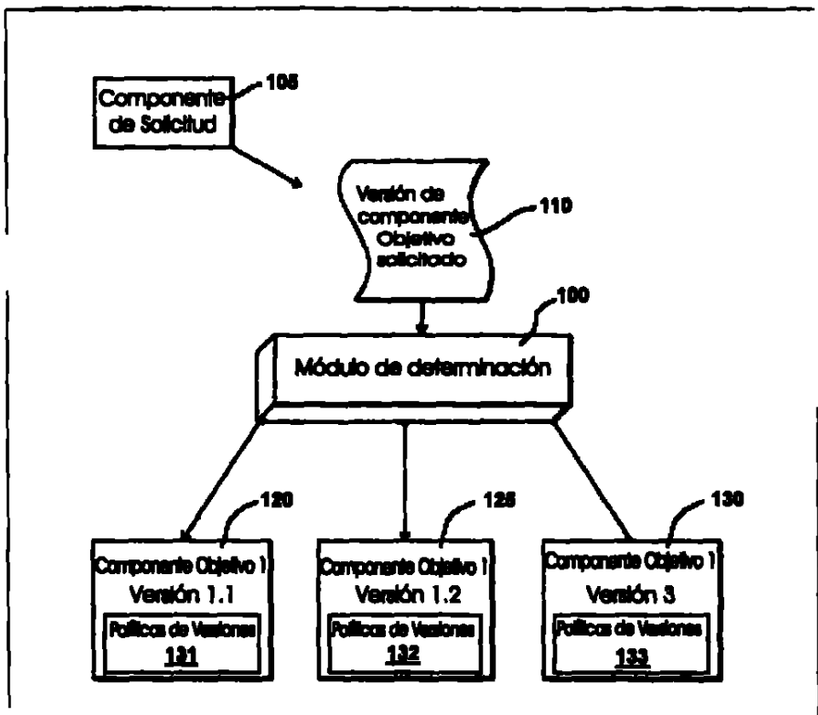

Componente de Solicitud
105
Versión de componente Objetivo solicitado
110
Módulo de determinación
100
Componente Objetivo 1 Versión 1.1
Políticas de Versiones 131
120
Componente Objetivo 1 Versión 1.2
Políticas de Versiones 132
125
Componente Objetivo 1 Versión 3
Políticas de Versiones 133
130

Figura 1

8

UNITED STATES PATENT APPLICATION

of

James S. Miller

Clemens Szyperski

Antony Scott Williams

John J. Rivard

Srivatsan Parthasarathy

C. Douglas Hodges

Patrick Dussud

William George Evans

and

Jonathan C. Hawkins

for

VERSIONING SUPPORT IN OBJECT-ORIENTED PROGRAMMING LANGUAGES AND TOOLS

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

VERSIONING SUPPORT IN OBJECT-ORIENTED PROGRAMMING LANGUAGES AND TOOLS

BACKGROUND OF THE INVENTION

1. The Field of the Invention

[0001] This invention relates to systems, methods, and computer program products for coordinating software components in a software environment.

2. Background and Relevant Art

[0002] Computerized, electronic systems are increasingly common, in part because such computerized systems automate much of what people previously had to perform manually. Accordingly, computerized systems have added a certain amount of efficiency to people's ability to perform tasks.

[0003] The process of generating computerized instructions (also referred to herein as "software" or "programs") for a computerized system is somewhat involved. Ordinarily, a software developer must first think of the desired functions or results that the program should perform, and then enter corresponding text-format instructions into an electronic text file, typically in the form of programming source code. In some cases, such as with interpreted programming languages (e.g., Javascript, Perl, etc.), a computerized system directly interprets entered text-format instructions, and performs the desired function. In other cases, such as with compiled programming languages (e.g., C# - pronounced "C sharp", C++, etc.), text-format instructions are first compiled into object or machine codes that the computerized system can execute.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0004] With more complicated programs, developers will sometimes implement the program's functionality in a number of interoperating "components". Generally speaking, components (or, program components) are sets of computer-executable instructions, much like a larger application program, although tending to be smaller and less complicated since they are typically geared toward providing one or few functions. Since a given component can run sometimes as an independent program, and can also communicate with other components, a more complicated program can also sometimes be referred to interchangeably as a "component". Furthermore, components can be referred to generally as either a "requesting component" or as a "target component", although such a designation may be arbitrary depending on which component or program is accessing the other.

[0005] In any case, a program designer can design one component on a computerized system to request access to any number of the computerized system's other components. Target components may include functions that provide basic information such as the user's name and age, or that provide more complicated information such as the user's level of use or sophistication with a given application program. Software components can also provide system functions such as executing a command to open a file, indicating communication protocols so that one component or program can interact with still other components, and so forth. Of course, one will appreciate that a large operating system can include many components that are configured to operate with multiple different programs, and vice versa.

[0006] Generally, a requesting component includes a reference to a target component. It may be that a requesting component references a specific version of the target component (a strict reference). Referencing a specific version of target

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

component may occur, for example, when a developer of the requesting component has prior knowledge of the target component and desires to make the requesting component expressly dependent on a specified version of the target component. For example, a requesting "component 1" may be configured to reference a target "version 1.1" of "component 3" to cause "component 1" to expressly depend on version 1.1" of "component 3. On the other hand, it may be that a requesting component references a target component that may or may not even exist when the requesting component was developed (a loose reference). Thus, a developer references a target component without prior knowledge of the target component. Accordingly, the requesting component may discover the existence of a version of the target component at run-time. For example, at run-time "component 1" may discover "version 2.1" of "component 3"

[0007] Unfortunately, there exist a number of disadvantages to implementing program components in the overall software design process, whether requesting components reference target components strictly or loosely . For example, when a user updates a target program that is referenced by one or more requesting components, the one or more requesting components may fail if the upgraded version of the target component turns out to be incompatible with the one or more requesting components. This problem can occur when the developer of the requesting component is not able to anticipate the number and type of changes that the developer of the relevant target component may implement in the future. By contrast, system policies prohibiting target component updates, or that prohibit component updates from overwriting prior versions of the target components, can result in systems that are quickly outdated, or that can become inefficient and bulky.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0008] Some attempts to overcome these problems have included system administrators trying to manage strict and loose references to different versions of target components in the same system. In such a scenario, a computerized system identifies the version number of a given target component when it is installed on the computer system, or when the target component is first run. The computerized system then stores the identified target component information along with other information about any other versions of the target component that have also been installed on the system. When a requesting component on the computerized system requests access of a target component, the computerized system then matches the requesting component to the requested version of the target component, as appropriate.

[0009] Unfortunately, there exist still a number of disadvantages to this type of system. For example, the only information available for a target component when it is installed on the system may be the version of the target component. However, the system cannot identify whether the particular version of the target component is an upgrade of a prior version of the target component. The system also cannot identify whether a developer intends to upgrade the particular version of the target component at some later point in time, since that information is unknown. This information, as well as other necessary operating parameters must be supplied by the system administrator.

[0010] For example, a system administrator must try to configure a system based on what little information about the given target component is available, or what the administrator expects, and then provide this information about the target component to the system when the given target component is installed, or first run. In particular, the system administrator must often provide access rules for different target components that indicate whether some requesting components are required to access specific

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

versions of other target components, and whether still other requesting components are allowed to access updated versions of other target components, and so forth. The system administrator must also provide any other information to the system as conflicts between versions of requesting and target components are realized. Thus, when a requesting component requests a given target component, the system typically grants access to the target component based on the version of the target requested, any versions of the target component stored on the system, and any other system administrator-supplied information.

[0011] One will appreciate, however, that this type of system that mixes strict and loose references to target components can be unduly complicated for system administrators. This is particularly true since system administrators are not always privy to what a third-party developer has in mind when the third-party developer is writing a given target component. Furthermore, system administrators cannot always anticipate whether certain target components are intended to be compatible with other versions or types of requesting components. This is particularly true for large systems where large numbers of requesting components are configured to access a wide variety of target components in any given number of ways.

[0012] Accordingly, an advantage in the art can be realized with systems, methods, and computer program products that allow present and future versions of requesting and target components to cooperate in a computerized system as configured. In particular, an advantage in the art can be realized with systems and methods that allow such component cooperation automatically, such that programs and components can continue to work effectively with little or no input from a system administrator.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

BRIEF SUMMARY OF THE INVENTION

[0013] The present invention solves one or more of the foregoing problems in the prior art with systems, methods, and computer program products that allow program developers to easily accommodate changes in components, modules, and operating systems without impairing program function. In particular, systems are disclosed that allow programs and components that access each other through static or dynamic references to compatibly coexist in an operating system.

[0014] In at least one exemplary implementation of the present invention, a determining module can receive a request to access a specified version of a target component from a requesting component. The request may include the versioning policy of the specified target component. Alternately, the determining module can identify the versioning policy of the specified target component. For example, a versioning policy can be included in a data field within the target component. Identification of the versioning policy and specified version can be done in response to the request, when the target component is installed, or when the target component is deployed on the computerized system. Other policies, such as, for example, component scope, can also be identified, as well as any system administrator-provided policies where appropriate. A requesting component is therefore granted access to an appropriate version of the target component primarily based on information contained within the request and contained within the target component.

[0015] In another exemplary implementation of the invention, the determining module receives a target component upgrade, and can identify a versioning policy that is associated with the target component and/or the requesting component. Based on the information provided in the versioning policy, the determining module can replace the

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

target component with the upgraded component, or can simply add the target component upgrade to the system so that the original and upgraded versions of the target component coexist. Hence, upgrades are processed for target components as appropriate for requesting components, such that any requesting component that accesses the target component will continue to access the original or prior version of the target component if necessary.

[0016] Additional features and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the invention. The features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0018] Figure 1 illustrates an example computer architecture for providing a requesting component with access to a target component, in accordance with the principles of the present invention;

[0019] Figures 2A illustrates an example computer architecture that receives newer versions of existing components in accordance with the principles of the present invention;

[0020] Figure 2B illustrates the example computer architecture of Figure 2A after a determination module has determined the versions of the components that are to be retained in accordance with the principles of the present invention;

[0021] Figure 3 illustrates an example computer architecture for stratifying component scope at different processing levels in accordance with the principles of the present invention;

[0022] Figure 4 illustrates an example flow chart of a method for providing component access in accordance with the principles of the present invention;

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0023] Figure 5A illustrates an example flow chart of a method for managing component upgrades in accordance with the principles of the present invention;

[0024] Figure 5B illustrates an example flow chart of a method for limiting component scope in accordance with the principles of the present invention; and

[0025] Figure 6 illustrates a suitable environment for practicing aspects of the present invention.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0026] The present invention extends to systems, methods, and computer program products that allow program developers to easily accommodate changes in components, modules, and operating systems without impairing program function. In particular, systems are disclosed that allow programs and components that access each other through static or dynamic references to compatibly coexist in an operating system. The embodiments of the present invention may comprise a special purpose or general-purpose computer including various computer hardware, as discussed in greater detail below.

[0027] Embodiments within the scope of the present invention also include computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer. By way of example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to carry or store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer.

[0028] When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a computer-readable medium. Thus, any such connection is properly termed a computer-readable medium. Combinations of the above should also be included within the scope of computer-readable media. Computer-executable instructions comprise, for example,

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions.

[0029] Figure 1 shows exemplary computer architecture for practicing an implementation of the present invention in which a determining module 100 receives one or more requests from a requesting component 105 to access another component or program, such as components 120, 125, and 130. For the purposes of this specification and claims, a "determining module" 100 can include any type of module having executable instructions configured, for example, for identifying a reference to a target component, which includes the name of the component and the originally intended version, and choosing an appropriate target component which will be used to satisfy the request. In some situations, this can include deciding that there is no such target component available, such that the determining module 100 would be configured to return an error. As will also be detailed hereinafter, the determining module 100 can also be configured to identify other information, such as which other components have already been made available for access within a computer system, process, or subprocess.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0030] In addition, a "component", for the purposes of this specification and claims will be understood to include any form of executable instructions that can be run on a computerized system, such as, for example, an interpreted text format file, as well as a file that has been compiled into machine-readable instructions. The term component, therefore, can include both larger application programs and systems that provide a large variety of functions, as well as smaller program and/or system components that provide other components or programs with specific functionality. Furthermore, although some

distinction will sometimes be made in this specification between "applications", "application programs", "programs", and components, the distinctions are merely one of convenience in order to clarify, usually, that one set of executable instructions are requesting, or are receiving a request, to access another component since each can be properly be referred to as a component. Hence, the terms "requesting component" and "target component" can include any of the foregoing executable instructions, as will be further detailed herein.

[0031] Embodiments of the presentation invention can access components that have been classified as "platform" or "library" components. "Platform" components are components that can be accessed by multiple other components or programs in a computerized system. Platform components are normally accessed only in the most recent form, or upgraded form, such that a requesting component may simply request the target component generally, or a minimum version of the target component, rather than request a specific version of the target component. Thus, the determining module may, for example, be configured to provide a version of a platform component other than the most recent version. In theory, platform components can be overwritten by a component upgrade when the upgrade is received, although there are reasons why this may not be done in practice. Platform components may also sometimes be referred to as "binary compatible" components. By contrast, library components are accessed by another component or program only if precisely the same version of the library component has been referenced.

[0032] As depicted in Figure 1, a determining module 100 can receive a request or declaration from a requesting component 105, to access a target component, such as component 120, 125, and 130. For the purposes of this disclosure and claims, the term

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

"target component" refers to a component for which access is sought by a requesting component. One will appreciate, however, that whether a component is a target component or a requesting component is primarily one of perspective, depending on which component requests access of the other. As such, the discussion as applied to target components in this specification and claims can apply equally to requesting components, and vice versa.

[0033] In any case, a requesting component 105 can initiate a component access request 110 through the determining module 100, where the request indicates that the requesting component 105 is configured to access a given version of a target component 120, 125, and 130. In some implementations, the request 110, can be a reference found in the source code of the requesting component 105 when the requesting component 105 is first installed on a given computerized system (not shown). Alternatively, a request 110 can be made by the requesting component 105 when the requesting component 105 requests access of a given version of a target component, such as components 120, 125, and 130, at run time.

[0034] A "versioning policy", for the purposes of this specification and claims includes any of a given set of properties that can be conveyed from a target component (e.g., 120, 125, 130) to a determining module 100. The versioning policy 131, 132, or 133 specifies whether the corresponding target component 120, 125, 130 can be used instead of a version of the given target component with a lower version number. The versioning policy can include additional information intended to be used by the determining module 100 to decide whether the target component can be used in a given configuration. Thus, the versioning policy 132 may specify that target component 125 (version 1.2) can be used when version 1.1 is requested. In some embodiments, the

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

versioning policy will be found in a predefined location within a target component. In other implementations, the versioning policy can be conveyed to the determining module 100 when the component is installed on the system, or when a first request is made to access a given component, and so forth.

[0035] Accordingly, a requesting component can request access to a target component by requesting a specific version of the target component, e.g., request 110 for "component 1" "version 1.1". If the requesting component 105 requests, or is configured to work with a specific version of a target component, determining module 100 can provide the requesting component 105 with access to a specific version of the component, depending on the versioning policy 131, 132, 133 that is present in the target component. As shown in Figure 1, for example, requesting component 105 requests "version 1" of "component 1" by sending a request 110. Hence, determining module 100 grants requesting component 105 access to "version 1.1" of "component 1" even though a more recent version of "component 1", e.g., "version 1.2" 125, exists in the system.

[0036] By contrast, in some embodiments, a request for one version of a component results in access to another (e.g., updated or more recent) version of a component. For example, request 100 may be a request to access "version 1.1" of "component 1". However, versioning policy 131 may indicate that "version 1.1" is a platform component (and thus a most recent version of "component 1" is to be provided in response to a request). Furthermore, in some implementations there may nevertheless be multiple versions of a given platform component on a system.

[0037] As such, the requesting component can also include information in its request that indicates the lowest possible version of the platform target component that

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

the requesting component 105 can accept. For example, it may be that requesting component 105 requests "version 1.4" of component 1" and that lower versions of "component 1" are not to be returned in response to the request. Accordingly, determining module 100 can provide requesting component 105 with access to "version 3" of "component 1", even though "version 1.1" and "version 1.2" are accessible.

[0038] When all accessible versions of a requested component have version designations lower than a specified requested version, determining module 100 can return an appropriate response (e.g.., an error message) to a requesting component. For example, when determining module 100 does not have access to "version 3" of "component 1", determining module 100 can send an error message to requesting component 105 in response to a request for "version 1.4 or higher" of component 1".

[0039] It may be that a version number includes two parts, a version and a servicing. Components that have a version number indicating an updated servicing are allowed to replace components that have version numbers indicating older servicing. Utilizing servicing values to facilitate component replacement is particularly advantageous for implementing minor changes that have a reduced likelihood of causing incompatibility with other components, for fixing bugs, or for fixing security issues whether related to library or platform components. That is, servicing values can facilitate "patching" a version of a component. For example, if target component 120 is identified as a library component (such that version 1.1. of the target component is not to be replaced), a developer can still update the target component 120 by updating (e.g., incrementing) a servicing value in the component's version number. Accordingly, the updated target component 120 would essentially be a different servicing of "version 1.1".

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0040] Figure 2A illustrates an example computer architecture that receives newer versions of existing components. That is, determination module 100 can receive upgrades to target components that are already resident at a corresponding computerized system. For example, determining module 100 can receive components 215 and 210 from a network service provider (not shown) connected to network 240. Determining module 100 can receive components 215 and 210 as the result of executing an installation program at the corresponding computerized system (or at the network service provider).

[0041] As depicted, components 210 and 215 include versioning policy information, such as, for example that upgraded component 210 is a "version 3" upgrade of "component 2", and that the component is a platform component. As well, component 215 can include information in the form of a versioning policy that the component 215 is library component, or that component 215 otherwise configured such that requesting components can be given access to the specific version represented by component 215.

[0042] In response to receiving the components 210 and 215, the determining module 100 determines whether to retain prior versions (which may be referred to as "side-by-side" updating) or replace prior versions (which may be referred to as "in-place" updating) of each received upgraded component. For example, as shown in Figure 2A components 220 and 235 are library and platform components respectively. More specifically, in response to receiving the component 215, the determining module 100 can identify that, since "component 1" is a library component, other programs or components may be configured to specifically access "version 1" of "component 1".

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

Accordingly, determining module 100 can determine that both component 215 and component 220 are to be retained.

[0043] More specifically, in response to receiving the component 210, the determining module 100 can identify that, since "component 2" is platform component, requesting programs and components will be given access to the most recent version of component 2. Accordingly, determining module 100 can determine that component 235 is to be replaced with component 210.

[0044] Figure 2B illustrates the example computer architecture of Figure 2A after a determination module has determined the versions of the components that are to be retained. As depicted in Figure 2B, both "version 1" (component 220) and "version 2" (component 215) of component 1 remain on the system (a side-by-side update). Also as depicted in Figure 2B, only "version 3" of "component 2" (component 210) remains on the system (an in-place update).

[0045] Figure 3 illustrates exemplary computer architecture for stratifying component scope at different processing levels in accordance with an implementation of the present invention. Stratification is based on component scope that applies to target components. By way of explanation and not of limitation, Figure 3 indicates three levels of scope, i.e., a "Machine" level 330, a "Process" level 340, and a "Sub-Process" level 350. One will appreciate, however, after reading this disclosure and claims that there can be greater or fewer numbers of levels, as appropriate. In particular, aspects of the invention allow a target component to supply a versioning policy that requires only one version of the target be made available at a given level (i.e. only one version on the entire machine or only one in a given process or only one in a given subprocess).

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0046] For example, a versioning policy that is associated with a given target component 300 can include a set of component scope. Referring briefly back to Figure 1, the component scope can indicate that a requesting component 105 must access the target component 300 at a given process level. As shown in Figure 3, for example, "component 1" "version 1" 300 is identified for machine-level access. Any requesting component installed in the system that requests access of the target component 300, must use "component 1" "version 1" since the target component 300 is configured for machine-level access. As with other versioning policy properties, this process level limitation can be indicated by the developer of a target component before the component is installed on a given system.

[0047] Component scope can also indicate larger or smaller scopes for a target component 300, 310, 315, 320, and 325. For example, "versioning policies" identified with a given component 310 can indicate that a given version of the target component 310 is required only within a certain process 342, 345, or sub-process 352, 355. As shown in Figure 3, for example, any requesting component 105 that requests access to a given version of a component 310 can do so within a process 342 without requiring other requesting components (on the system) to use the same target component in other processes 315. As such, component 310 can be used in process 342, while component 315 can be used in process 345. Furthermore, when process A 342 has not selected a particular version, sub-process 350, which depends from process 340, can use different versions of component 310, such as components 320 and 325. This level of granular access to different components can therefore be indicated when the given component is developed, rather than by a system administrator when the given component is installed on the system. The determining module 100 can combine any identified component

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

scope for each target or requesting component to provide a requesting component with appropriate target component access.

[0048] Accordingly, identification of an appropriate version of a target component can be based on other policies, such as, for example, component. A determining module can therefore identify an appropriate version of a target component based on any identified policy, such as the versioning policy and component scope of the specified target component, as well as any other system administrator-provided policies where appropriate

[0049] The present invention may also be described in terms of methods comprising functional steps and/or non-functional acts. Figures 4, 5A, and 5B illustrate exemplary flow charts for allowing component access by other programs or components in a computerized system. The methods of Figures 4, 5A, and 5B will be discussed with respect to the modules of the programs illustrated in the preceding Figures.

[0050] Figure 4 illustrates an example flow chart of a method for providing component access in accordance with an implementation of the present invention. The method of Figure 4 includes an act 400 of receiving a request for a version of a target component. Act 400 can include receiving a request to access a specified version of a target component, the request being received from a requesting component. For example, a requesting component 105 can request access of a target component, such as component 120, 125, and 130 through a determining module 100. It may also be that the request includes the versioning policy of the version of the target component.

[0051] The method also includes a functional result-oriented step 440 of providing an appropriate target component. Step 440 can include any number of corresponding acts for implementing the present invention. However, as depicted in Figure 4, step 440

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

includes an act 410 of identifying a versioning policy. Act 410 can include identifying a versioning policy of the specified version of the target component. If a versioning policy was included in the request, the determine module 100 may identify such a n included versioning policy. Alternately, the determine module 100 can refer to one or more versions of the target component and identify versioning policies stored in the one or more versions of the target component. For example, the determining module 100 can identify that multiple versions of a component such as a "version 1" 120 and a "version 2" 125 of the same "component 1" exist on the system, each having a corresponding versioning policy 131, 132, and 133. A software developer can include a versioning policy in target components 120, 125, and 130, etc., such that a determining module 100 identifies the versioning policy upon compiling, installing, and or running the developed program or component.

[0052] Step 440 also includes an act 430 of providing an appropriate version of the target component. Act 430 can include identifying an appropriate version of the target component based on the versioning policy of the specified target component. For example, the determining module 100 can provide a requesting component 105 with a specific version of the requested target component (a library component), such as component 120. Alternatively, the determining module 100 can provide the requesting component 105 with a more recent version of a component (a platform component), such as component 130.

[0053] Figure 5A illustrates an example flow chart of a method for managing component upgrades in accordance with an implementation of the present invention. The method of Figure 5A can be implemented such that a requesting component that accesses the target component continues to operate effectively after the target

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

component has been upgraded. As illustrated, the method depicted in Figure 5A includes an act 500 of receiving a component update. Act 500 can also include identifying that a target component is accessed by a requesting component. For example, referring back to Figure 1, a determining module 100 may be linked to, or contain, a registry or database that, upon installation of a requesting component 105, identifies that the installed program or component or component 105 is configured to access a specific version of a target component such as a "version 1" 120 of "component 1". This determining module 100 can gain this information based on any versioning policy contained in the installed program, as well as contained within any components that program is configured to access.

[0054] The method depicted in Figure 5A includes an act 510 of identifying a versioning policy. Act 510 can include identifying a versioning policy in a prior version of the target component, and an updated version of the target component. For example, the determining module 100 identifies the versioning policy in any of the target components 120, 125, and 130, which, as previously described, can indicate the version of the target component 120, 125, and 130, and can indicate that the target component is intended to be a platform or a library component.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0055] The method depicted in Figure 5A further includes an act 520 of adding the component update to the system based at least in part on the versioning policy. Act 520 can include deleting a prior version of the target component and/or adding the updated version of the target component based on the identified versioning policy. For example, if a specific prior version 220 of the component is required for access by another program or component, such as if the component is a library component, the determining module 100 will not overwrite the prior version 220. The determining

module 100 will simply add the new version 215 of the component such that programs or components that request a new version 215 of the component can access it. Similarly, programs or components that require the prior version 220 of the component may also access that as well, which preserves the integrity of the requesting program or component. By contrast, if no program or component is identified as requiring a specific version of a given component (a platform component), the determining module 100 can simply overwrite the prior version 235 of the component with the recent version 210 of the component.

[0056] Figure 5B illustrates an example flow chart of a method for providing component access at one or more process levels in accordance with an implementation of the present invention. The method of Figure 5B can be implemented to organize one or more target components such that access to the one or more target components is limited. The method depicted in Figure 5B includes an act 550 of identifying a versioning policy. Act 550 can include identifying a versioning policy in a target component. For example, a determining module 100 can receive a component access request from a requesting component, and can identify a versioning from within a received upgrade 215, 210, within an existing target component 220, 225, , and so forth. As previously described, the versioning policy can help the system identify a version number of the target component, as well as whether the target component 220, 225, and 230 is intended to be a library or platform component.

[0057] The method depicted in Figure 5B includes an act 560 of identifying a component scope associated with the component. Act 560 can include identifying a component scope associated with the target component where the component scope identifies a property that is associated with a requesting component that can be

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

configured to access the target component. For example, either the target component or the requesting program or component can be associated with a specific component scope that indicates that a version of the target component can be accessed at one of a machine layer, a sub-process layer, and so forth. One will appreciate, however, as described herein, that there can be variety of levels in which component access can be limited, depending on a developer's preferences.

[0058] The method depicted in Figure 5B includes an act 570 of allowing target component access based on the component scope and the versioning policy. Act 570 can include allowing at least one of the one or more requesting components to access the target component based on the access property associated with a requesting component and the identified versioning policy. For example, if one or more programs or components 300 are indicated for machine-wide processes 330, only that version of the component 300 will be available to any given requesting component at any given process level. By contrast, if the target component is identified with process level access, the determining module 100 can allow other requesting components to access different versions of the same target component for a given corresponding process or sub-process, as appropriate.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0059] Figure 6 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by computers in network environments. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Computer-executable instructions, associated

data structures, and program modules represent examples of the program code means for executing steps of the methods disclosed herein. The particular sequence of such executable instructions or associated data structures represents examples of corresponding acts for implementing the functions described in such steps.

[0060] Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including personal computers, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by local and remote processing devices that are linked (either by hardwired links, wireless links, or by a combination of hardwired or wireless links) through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0061] Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including personal computers, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where local and remote processing devices perform tasks and are linked (either by hardwired links, wireless links, or by a combination of hardwired or wireless links) through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0062] With reference to Figure 6, an exemplary system for implementing the invention includes a general-purpose computing device in the form of a conventional computer 620, including a processing unit 621, a system memory 622, and a system bus 623 that couples various system components including the system memory 622 to the processing unit 621. The system bus 623 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. The system memory includes read only memory (ROM) 624 and random access memory (RAM) 625. A basic input/output system (BIOS) 626, containing the basic routines that help transfer information between elements within the computer 620, such as during start-up, may be stored in ROM 624.

[0063] The computer 620 may also include a magnetic hard disk drive 627 for reading from and writing to a magnetic hard disk 639, a magnetic disc drive 628 for reading from or writing to a removable magnetic disk 629, and an optical disc drive 630 for reading from or writing to removable optical disc 631 such as a CD ROM or other optical media. The magnetic hard disk drive 627, magnetic disk drive 628, and optical disc drive 630 are connected to the system bus 623 by a hard disk drive interface 632, a magnetic disk drive-interface 633, and an optical drive interface 634, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-executable instructions, data structures, program modules and other data for the computer 620. Although the exemplary environment described herein employs a magnetic hard disk 639, a removable magnetic disk 629 and a removable optical disc 631, other types of computer readable media for storing data can be used, including magnetic cassettes, flash memory cards, digital versatile disks, Bernoulli cartridges, RAMs, ROMs, and the like.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0064] Program code means comprising one or more program modules may be stored on the hard disk 639, magnetic disk 629, optical disc 631, ROM 624 or RAM 625, including an operating system 635, one or more application programs 636, other program modules 637, and program data 638. A user may enter commands and information into the computer 620 through keyboard 640, pointing device 642, or other input devices (not shown), such as a microphone, joy stick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 621 through a serial port interface 646 coupled to system bus 623. Alternatively, the input devices may be connected by other interfaces, such as a parallel port, a game port or a universal serial bus (USB). A monitor 647 or another display device is also connected to system bus 623 via an interface, such as video adapter 648. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

[0065] The computer 620 may operate in a networked environment using logical connections to one or more remote computers, such as remote computers 649a and 649b. Remote computers 649a and 649b may each be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically include many or all of the elements described above relative to the computer 620, although only memory storage devices 650a and 650b and their associated application programs 636a and 636b have been illustrated in Figure 6. The logical connections depicted in Figure 6 include a local area network (LAN) 651 and a wide area network (WAN) 652 that are presented here by way of example and not limitation. Such networking environments are commonplace in office-wide or enterprise-wide computer networks, intranets and the Internet.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

[0066] When used in a LAN networking environment, the computer 620 is connected to the local network 651 through a network interface or adapter 653. When used in a WAN networking environment, the computer 620 may include a modem 654, a wireless link, or other means for establishing communications over the wide area network 652, such as the Internet. The modem 654, which may be internal or external, is connected to the system bus 623 via the serial port interface 646. In a networked environment, program modules depicted relative to the computer 620, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing communications over wide area network 652 may be used.

[0067] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes that come within the meaning and range of equivalency of the claims are to be embraced within their scope.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

CLAIMS

We claim:

1. In a computerized system that includes one or more program components including one or more requesting components that can request to access one or more target components, a method of providing a requesting component with access to an appropriate version of a target component, comprising the acts of:

receiving a request to access a specified version of a target component, the request being received from a requesting component;

identifying a versioning policy of the specified version of the target component;

identifying an appropriate version of the target component based on the versioning policy of the specified target component;

providing the requesting component with access to the appropriate version of the target component.

2. The method as recited in claim 1, wherein the requested version of the target component is one of a library component and a platform component.

3. The method as recited in claim 1, wherein identifying an appropriate version of the target component comprises identifying a more recent version of the target component in response to a request for an earlier version of the target even though the more recent version and the earlier version are both accessible to the computerized system.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

4. The method as recited in claim 3, identifying a more recent version of the target component in response to a request for an earlier version of the target even though the more recent version and the earlier version are both accessible to the computerized system comprises identifying a more recent version of a platform component even though an earlier version of the platform component remained on the system when the more recent version was received at the computerized system.

5. The method as recited in claim 1, wherein the versioning policy of the specified version of the target component is identified when the target component is one or more of compiled, configured, installed, and run on the computerized system.

6. The method as recited in claim 1, wherein version information that identifies the specified target component is stored in the requesting component when the requesting component is one or more of compiled, configured, installed, and run on the computerized system.

7. The method as recited in claim 1, further comprising:

identifying one or more requesting components that are able to access a prior version of the target component;

identifying that none of the one or more requesting components are configured to access the prior version of the target component; and

deleting the prior version of the target component.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

8. The method as recited in claim 1, wherein the appropriate version of the target component is the version of the target component that was requested.

9. The method as recited in claim 1, wherein the appropriate version of the target component is different from the version of the target component that was requested.

10. The method as recited in claim 9, wherein target component access is provided to the requesting component through a determining module.

11. The method as recited in claim 10, wherein the availability of one or more of the prior version of the target component and the more recent version of the target component is identified by a determining module when the one or more of the prior version of the target component and the more recent version of the target component is received by the computerized system.

12. The method as recited in claim 1, wherein the versioning policy is inserted into computer-executable instructions in the target component prior to one of installing, configuring, and executing the target component on the computerized system.

13. The method as recited in claim 1, wherein the versioning policy is further identified in any version of the target component.

14. The method as recited in claim 13, wherein the versioning policy identifies that any of the prior version of the target component and the more recent version of the

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

target component is configured to be accessed by a specific version of the requesting component.

15. The method as recited in claim 1, further comprising identifying a compcnent scope that is associated with the target component.

16. The method as recited in claim 15, wherein access to the specified version of the target component is further based on one of the identified component scope associated with the target component, and a target component scope supplied by a system administrator.

17. The method as recited in claim 16, wherein the identified component scope specifies that access to the specified version of the target component is provided in one or more of a machine level, a process level, and a sub-process level.

18. The method as recited in claim 1, wherein the requested target component is a library component, the method further comprising identifying a servicing value associated with the requested target component.

19. The method as recited in claim 1, wherein identifying an appropriate version of the target component comprising identifying an updated version of a library component based on the identified versioning policy and the identified servicing value.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

20. In a computerized system that includes one or more program components including one or more requesting components that can request to access one or more target components, a method of providing a requesting component with access to an appropriate version of a target component, comprising:

an act of receiving a request to access a specified version of a target component, the request being received from a requesting component;

a step for allowing access to an appropriate version of the requested target component such that the requesting component accesses the appropriate target component as it has been configured to do so, and such that the requesting component does not fail when requesting access to a component that has been upgraded.

21. The method as recited in claim 20, wherein the step for allowing access to an appropriate version of the requested target component comprises the corresponding acts of:

identifying a versioning policy of the specified version of the target component;

receiving a request to access a specified version of a target component, the request begin received from a requesting component;

providing the requesting component with access to the appropriate version of the target component.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

22. In a computerized system that includes one or more program components including one or more requesting components that can request to access one or more target components, a method of upgrading a target component such that a requesting component that accesses the target component continues to operate effectively after the target component has been upgraded, comprising the acts of:

identifying that a requesting component is configured to access a target component;

identifying a versioning policy in at least an existing version of the target component and a previously installed version of the target component; and

identifying which versions of the target component should remain on the system based on any identified versioning policy corresponding to at least the existing version of the target component and the previously installed version of the target component.

23. The method as recited in claim 22, further comprising receiving an updated version of the target component over a network from a network service provider.

24. The method as recited in claim 22, wherein, if the versioning policy indicates that the requesting component is a library component, adding the existing version of the target component to the system without removing the previously installed version of the target component.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

25. The method as recited in claim 22, wherein, if the versioning policy indicates that the requesting component is a platform component, overwriting the previously installed version of the target component with the existing version of the target component.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

26. In a computerized system including one or more requesting components that are configured to access one or more source components, a computer program product having computer-executable instructions stored thereon that, when executed, cause the computerized system to perform a method of providing a requesting component with access to an appropriate version of a target component, comprising the acts of:

receiving a request to access a specified version of a target component, the request being received from a requesting component;

identifying a versioning policy of the specified version of the target component;

identifying an appropriate version of the target component based on the versioning policy of the specified target component;

providing the requesting component with access to the appropriate version of the target component.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

27. In a computerized system including one or more requesting components that are configured to access one or more source components, a computer program product having computer-executable instructions stored thereon that, when executed, cause the computerized system to perform a method of upgrading a target component such that a requesting component that accesses the target component continues to operate effectively after the target component has been upgraded, comprising the acts of:

identifying that a requesting component is configured to access a target component;

identifying a versioning policy in at least an existing version of the target component and a previously installed version of the target component; and

identifying which versions of the target component should remain on the system based on any identified versioning policy corresponding to at least the existing version of the target component and the previously installed version of the target component.

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

ABSTRACT OF THE DISCLOSURE

A versioning policy included in a target component indicates how the target component is to be accessed, for example, either as a library component or a platform component. A component may be designated as a library component when it is not versioned in a binary compatible manner. When other components request such a component they receive specifically the version of the component they requested. On the other hand, a component may be designated as a platform component when it is versioned in a binary compatible manner. When other components request such a component they may receive the latest upgraded version of the component requested instead. Thus, access to an appropriate version of the component (even a version differing from the requested version) is facilitated. Other embodiments include mechanisms for stratifying component scope based on different processing levels.

W:\13768\493\MJF0000000372V001.doc

WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111

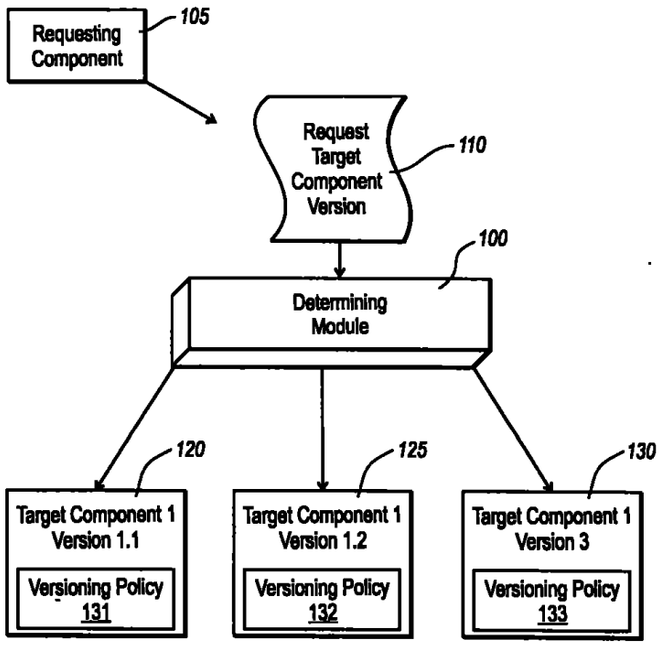

Requesting Component
105
Request Target Component Version
110
100
Determining Module
120
125
130
Target Component 1 Version 1.1
Versioning Policy 131
Target Component 1 Version 1.2
Versioning Policy 132
Target Component 1 Version 3
Versioning Policy 133

Fig. 1

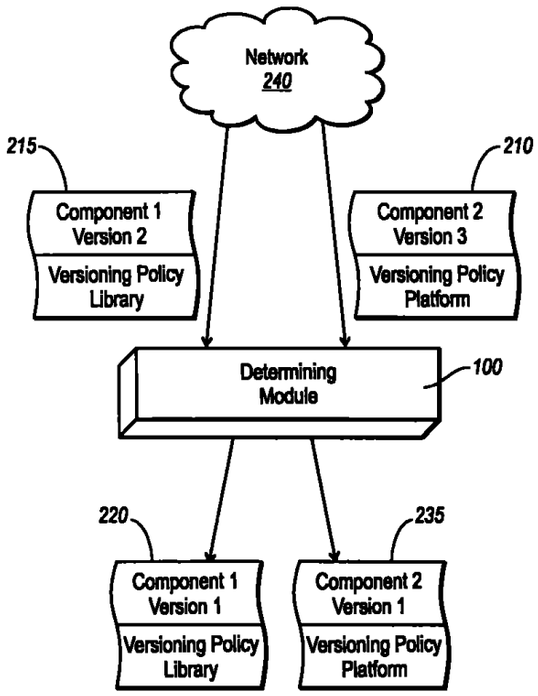

Network
240
215
210
Component 1
Version 2
Versioning Policy
Library
Component 2
Version 3
Versioning Policy
Platform
Determining
Module
100
220
235
Component 1
Version 1
Versioning Policy
Library
Component 2
Version 1
Versioning Policy
Platform

Fig. 2A

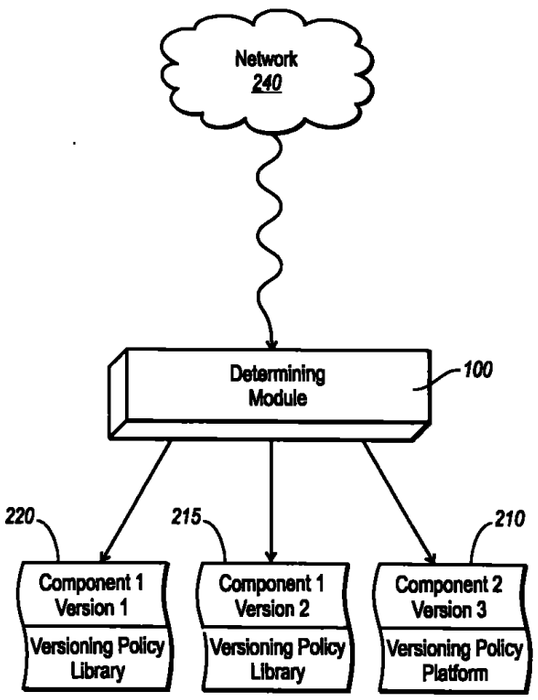

Network
240
Determining
Module
100
220
215
210
Component 1
Version 1
Versioning Policy
Library
Component 1
Version 2
Versioning Policy
Library
Component 2
Version 3
Versioning Policy
Platform

Fig. 2B

48

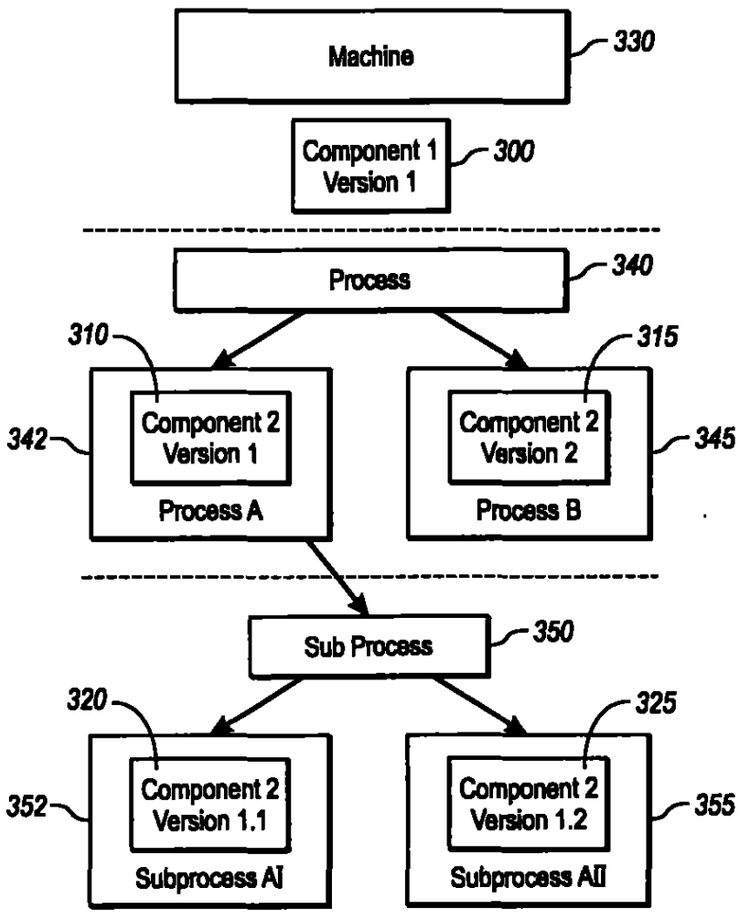

Machine
330
Component 1
Version 1
300
Process
340
310
315
Component 2
Version 1
Component 2
Version 2
342
345
Process A
Process B
Sub Process
350
320
325
Component 2
Version 1.1
Component 2
Version 1.2
352
355
Subprocess AI
Subprocess AII

Fig. 3

49

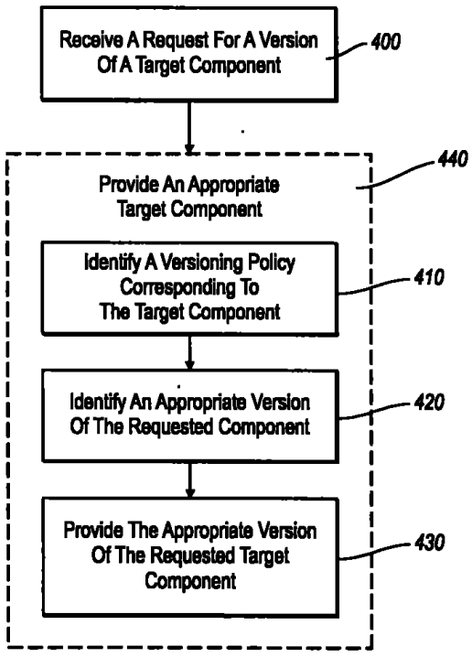

Receive A Request For A Version Of A Target Component
400
Provide An Appropriate Target Component
440
Identify A Versioning Policy Corresponding To The Target Component
410
Identify An Appropriate Version Of The Requested Component
420
Provide The Appropriate Version Of The Requested Target Component
430

Fig. 4

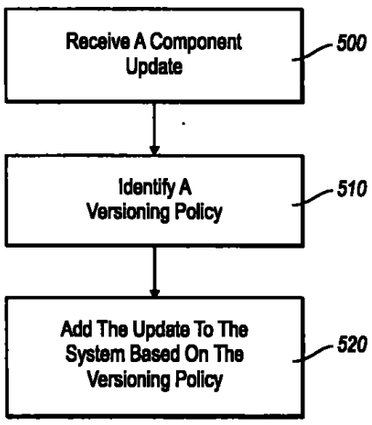

Receive A Component Update
500
Identify A Versioning Policy
510
Add The Update To The System Based On The Versioning Policy
520

Fig. 5A

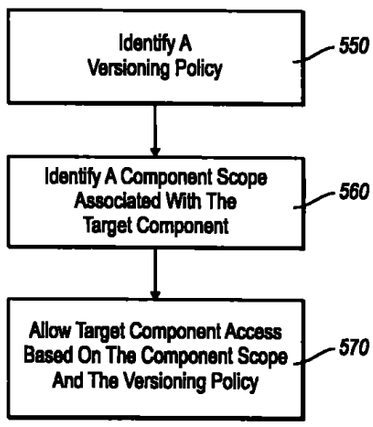

Identify A Versioning Policy
550
Identify A Component Scope Associated With The Target Component
560
Allow Target Component Access Based On The Component Scope And The Versioning Policy
570

Fig. 5B

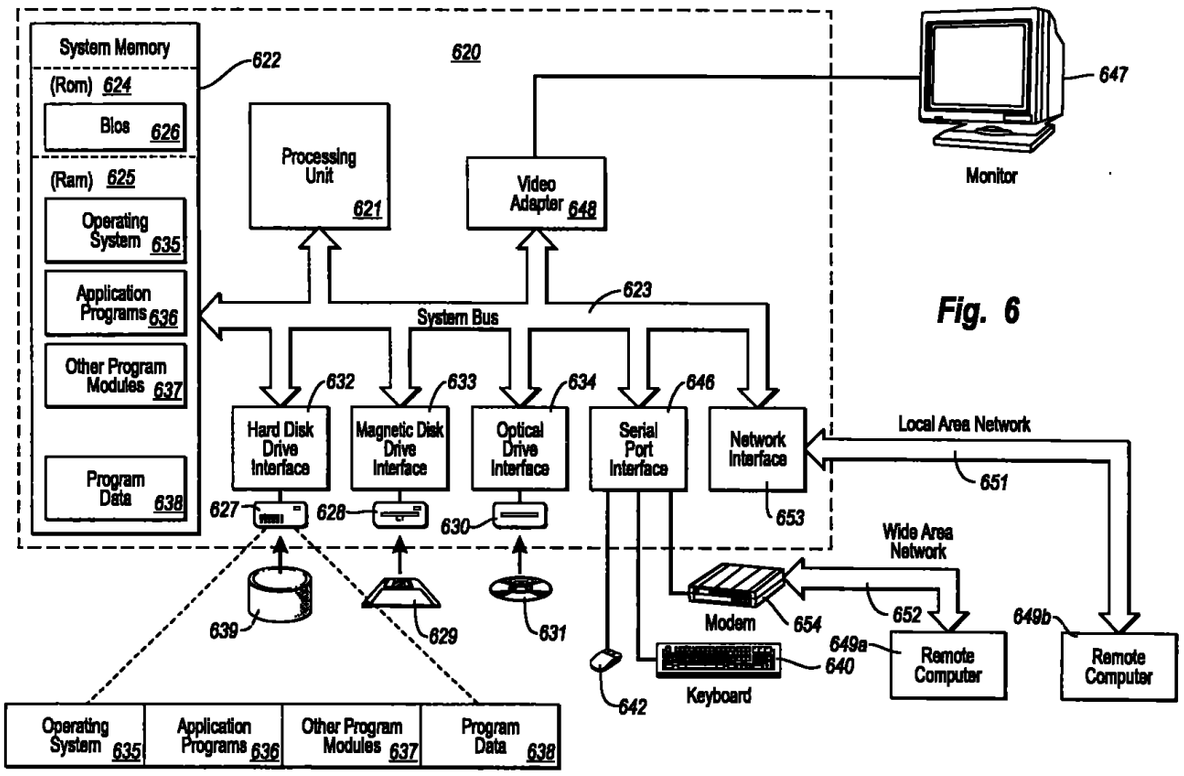

System Memory
622
620
(Rom) 624
Bios 626
(Ram) 625
Operating System 635
Application Programs 636
Other Program Modules 637
Program Data 638
Processing Unit 621
Video Adapter 648
647
Monitor
623
System Bus
632
633
634
646
Hard Disk Drive Interface
Magnetic Disk Drive Interface
Optical Drive Interface
Serial Port Interface
Network Interface
Local Area Network
651
653
627
628
630
639
629
631
Wide Area Network
652
Modem
654
649a
649b
640
Keyboard
642
Remote Computer
Remote Computer
Operating System 635
Application Programs 636
Other Program Modules 637
Program Data 638

Fig. 6

SOPORTE DE VERSIONES EN LENGUAJES Y HERRAMIENTAS DE PROGRAMACIÓN ORIENTADAS A OBJETO

ANTECEDENTES DE LA INVENCIÓN

1. El campo de la Invención

Esta invención se relaciona con sistemas, métodos y productos de programa de computador para coordinar componentes de software en un ambiente de software

2. Antecedentes de la Técnica Relevante

Los sistemas electrónicos computarizados son crecientemente comunes, en parte porque tales sistemas computarizados automatizan mucho de lo que las personas tenían previamente que desarrollar manualmente. De acuerdo con esto los sistemas computarizados han agregado una cierta cantidad de eficiencia a la capacidad de las personas para desarrollar tareas.

El proceso de generar instrucciones computarizadas (también denominadas aquí como "software" o "programas")

53

para un sistema computarizado está algo involucrado. Corrientemente un desarrollador de software debe primero pensar las funciones o resultados deseados que el programa debe desarrollar y luego ingresar las instrucciones de formato de texto correspondientes dentro de un archivo de texto electrónico, típicamente en la forma de un código fuente de programación. En algunos casos tal como los lenguajes de programación interpretados (por ejemplo Javascript, Perl, etc.), un sistema computarizado interpreta directamente las instrucciones de formato de texto ingresadas y desarrolla la función deseada. En otros casos tal como con lenguajes de programación compilados (por ejemplo C#- pronunciado "C sharp", C++, etc.,) las descripciones de formato de texto son primero compiladas en códigos objetos o de máquina que el sistema computarizado pueda ejecutar.

Con programas más complicados los desarrolladores algunas veces implementaran la funcionalidad del programa en un número de "componentes" interoperantes. Hablando en términos generales los componentes (o, los componentes de programa) son un conjunto de instrucciones ejecutables en computadora, como un programa de aplicación mayor, aunque tendiendo a ser más pequeños y menos complicados en razón

SH

54

a que ellos típicamente están armados hacia suministrar una o pocas funciones. En razón que un componente dado pueda correr algunas veces como un programa independiente y también puede comunicarse con otros componentes, un programas más complicado también puedes ser algunas veces ser denominado intercambiablemente "componente" adicionalmente, los componentes pueden ser denominados de manera general como un "componente solicitante" o como "componente objetivo", aunque tal designación puedes ser arbitraría dependiendo de cual componente de programa está accesando el otro.

En cualquier caso, un diseñador de programas puede diseñar un componente sobre un sistema computarizado para solicitar acceso a cualquier número de los sistemas computarizados de los otros componentes. Los componentes objetivo pueden incluir funciones que suministran la información básica tal como nombre de usuario y edad, o que suministran información más complicada tal como el nivel de usuario de uso o sofisticación con un programa de aplicación dado. Los componentes de software también pueden suministrar funciones de sistema tal como ejecutar un comando para abrir un archivo, indicando los protocolos de comunicación de tal forma que un componente

55

o programa puede interactuar con aún otros componentes, y así sucesivamente. Por supuesto, uno apreciará que un sistema operativo grande puede incluir muchos componentes que son configurados para operar con múltiples programas diferentes, y viceversa.

De manera general, un componente solicitante incluye una referencia a un componente objetivo. Puede ser que un componente solicitante referencie una versión específica del componente objetivo (una referencia estricta). Referenciar una versión específica del componente objetivo puede ocurrir, por ejemplo, cuando un desarrollador del componente solicitante tiene conocimiento previo del componente objetivo y desea elaborar el componente solicitante dependiendo expresamente de la versión especificada del componente objetivo. Por ejemplo, "un componente 1" solicitante puede ser configurado para referenciar una "versión 1.1" objetivo del "componente 3" para hacer que el "componente 1" dependa expresamente de la versión 1.1 del "componente 3". De otra parte puede ser que el componente solicitante referencia un componente objetivo que pueda no existir aun cuando el componente solicitante fuera desarrollado (una referencia suelta). De esta manera, un desarrollador

56

referencia un componente objetivo sin conocimiento previo del componente objetivo. De acuerdo con esto, el componente solicitante puede descubrir la existencia de una versión del componente objetivo en tiempo de ejecución. Por ejemplo, en el tiempo de ejecución "el componente 1" puede descubrir la "versión 2.1" del "componente 3"

Desafortunadamente, existe un número de desventajas para implementar componentes de programa en el proceso de diseño de software total, si los componente solicitantes referencia componentes objetivos estrictamente o sueltamente. Por ejemplo, cuando un usuario actualiza un programa objetivo que es referenciado por uno o más componentes solicitantes, el uno o más componentes solicitantes pueden fallar si la versión actualizada del componente objetivo resulta ser incompatible con uno o más componentes solicitantes. Ese problema puede ocurrir cuando el desarrollo del componente solicitante no es capaz de anticipar el número y el tipo de cambios que el desarrollador del componente objetivo relevante podría implementar en el futuro. Como contraste las políticas del sistema que prohíben las actualizaciones de los componentes objetivo o que prohíbe las actualizaciones

57

57

del componente de sobrescribir versiones anteriores de los componentes objetivo, puede dar como resultado sistemas que son rápidamente desactualizados o que se pueden volver ineficientes y voluminosos.

Algunos intentos de solucionar estos problemas han incluido administradores de sistema que tratan de manejar referencias estrictas y sueltas de diferentes versiones de los componentes objetivo en el mismo sistema. En tal escenario, un sistema computarizado identifica el número de versión de un componente objetivo dado cuando este se instala en el sistema de computador, o cuando el componente objetivo es ejecutado primero. El sistema computarizado almacena entonces la información de componente objetivo identificada junto con otra información acerca de cualesquiera otras versiones del componente objetivo que han sido instaladas en el sistema. Cuando el componente solicitante en el sistema computarizado solicita acceso de un componente objetivo, el sistema computarizado hace casar el componente solicitante con la versión solicitada con el componente objetivo, según sea apropiado.

58

58.

Desafortunadamente, existe un número de desventajas de este tipo de sistema. Por ejemplo, la sola información disponible para un componente objetivo cuando éste se instala en el sistema puede ser la versión del componente objetivo. Sin embargo, el sistema no puede identificar si la versión particular del componente objetivo es una actualización de una versión previa del componente objetivo. El sistema tampoco puede identificar si un desarrollador pretende actualizar la versión particular del componente objetivo en algún momento posterior, en razón a que esa información es desconocida. Esta información, así como también otros parámetros de operación necesarios debe ser suministrada por el administrador del sistema.

Por ejemplo, un administrador del sistema debe tratar de configurar un sistema basado en que la información pequeña se encuentra disponible acerca del componente objetivo dado, o que espera el administrador, y entonces suministra esta información acerca del componente objetivo al sistema cuando el componente objetivo dado se instala o se ejecuta primero. En particular, el sistema de administrador debe al menos suministrar reglas de acceso para diferentes componentes objetivo que indica si

59

algunos componentes solicitantes son requeridos para acceder a las versiones especificas de los otros componentes objetivo y si aún otros componentes solicitantes se les permite acceder a las versiones actualizadas de otros componentes objetivo, y así sucesivamente. El administrador del sistema también debe suministras cualquier otra información al sistema como los conflictos que se detecten entre las versiones de los componentes solicitantes y objetivo. Así, cuando el componente solicitante requiere un componente objetivo dado, el sistema típicamente da acceso el componente objetivo con base en la versión del objetivo solicitado, cualesquiera versiones del componente objetivo almacenadas en el sistema y cualquiera otra información suministrada por el administrador del sistema.

Uno apreciará, sin embargo, que este tipo de sistema que mezcla referencias estrictas y sueltas a los componentes objetivo puede ser indudablemente complicado para los administradores del sistema. Esto es particularmente cierto en razón a que los administradores del sistema no son siempre privados a lo que un tercer desarrollador tiene en mente cuando el tercer desarrollador está escribiendo un componente objetivo dado. Adicionalmente

60

los administradores de sistema no pueden siempre anticipar si ciertos componentes objetivo están destinados a ser compatibles con otras versiones o tipos de componentes solicitantes. Esto es particularmente cierto para grandes sistemas donde grandes números de componentes solicitantes son configurados para acceder una amplia variedad de componentes objetivo en cualquier número dado de formas.

De acuerdo con esto una ventaja en la técnica puede ser enterarse de sistemas, métodos y productos de programa de computador que le permitan a las versiones presentes y futuras de componentes solicitantes y objetivo cooperar en un sistema computarizado como se configuró. En particular, una ventaja en la técnica puede ser enterase de sistemas y métodos que permitan automáticamente tal cooperación componente, de tal forma que los probamos y los componentes pueden continuar trabajando efectivamente con poca o ninguna entrada de un administrador de sistema.

61

BREVE RESUMEN DE LA INVENCIÓN

La presente invención resuelve uno o más de los problemas anteriores en la técnica anterior con sistemas, métodos, y productos de programa de computador que le permite a los desarrolladores de programas acomodar fácilmente cambios en componentes, módulos, y en sistemas operativos sin dañar la función del programa. En particular se describen sistemas que le permiten a los programas y componentes que accede uno al otro a través de referencias estáticas o dinámicas para coexistir compatiblemente en un sistema operativo.

En al menos una implementación de ejemplo de la presente invención un módulo determinante puede recibir una solicitud para acceder a una versión específica de un componente objetivo desde un componente solicitante.

La solicitud puede incluir la política de versiones del componente objetivo especificado. Alternativamente el módulo determinante puede identificar la política de versiones del componente objetivo específico. Por ejemplo, una política de versiones puede ser incluida en el campo de datos dentro del componente objetivo. La

6P

62

identificación de la política de versiones y la versión específica puede ser hecha en respuesta a una solicitud, cuando el componente objetivo se instala, o cuando el componente objetivo es desplegado en el sistema computarizado. Otras políticas, tales como, por ejemplo, el alcance del componente, también pueden ser identificadas, así como también cualquiera de las políticas suministradas por el administrador del sistema cuando sea apropiado. Por lo tanto se le otorga acceso a un componente solicitante a una versión apropiada del componente objetivo primariamente con base a la información contenida dentro de la solicitud y contenida dentro del componente objetivo.

En otra implementación de ejemplo de la invención, el módulo determinante recibe un componente objetivo actualizado, y puede identificar una política de versiones que está asociada con el componente objetivo y/o el componente solicitante. Con base en la información suministrada en la política de versiones, el módulo determinante puede reemplazar el componente objetivo con el componente actualizado o puede simplemente agregar el componente objetivo actualizado al sistema de tal forma que las versiones original y actualizada del componente

63

objetivo coexistan. De esta manera las actualizaciones son procesadas por los componentes objetivo según se ha apropiado para los componentes solicitantes, de tal forma que cualquier componente solicitante que aceda al componente objetivo continuará accediendo a la versión original o anterior del componente objetivo si es necesario.

Las características y ventajas adicionales de la presente invención serán establecidas en la descripción que sigue y en partes serán obvias de la descripción o pueden ser aprendidas por la práctica de la invención. Las características y ventajas de la invención pueden ser comprendidas y obtenidas por medio de los instrumentos y las combinaciones particularmente puntualizadas en las reivindicaciones finales. Estas y otras características de la presente invención se volverán más evidentes a partir de la descripción siguiente y las reivindicaciones finales, o pueden ser aprendidas por la práctica de la invención como se establece continuación.

64

BREVE DESCRIPCIÓN DE LOS DIBUJOS

Con el fin de describir la manera en que pueden ser obtenidas las ventajas y características anteriormente mencionadas y otras de la invención, una descripción más particular de la invención brevemente descrita anteriormente será suministrada como referencia a las modalidades específicas de ésta que son ilustradas en los dibujos finales. El entendimiento de que estos dibujos solo describen modalidades típicas de la invención y no deben ser por lo tanto consideradas como limitantes de su alcance la invención será descrita y explicada con especificidad y detalles adicionales a través del uso de los dibujos que los acompañan en los cuales:

La figura 1 ilustra una arquitectura de computadora de ejemplo para suministrar un componente solicitante con acceso a un componente objetivo, de acuerdo con los principios de la presente invención;

La figura 2A ilustra una arquitectura de computadora de ejemplo que reciben presiones más nuevas de componentes existentes de acuerdo con los principios de la presente invención;

65

La figura 2B ilustra una arquitectura de computadora de ejemplo de la figura 2A después de que un módulo de determinación ha determinado las versiones de los componentes que tienen que ser retenidas de acuerdo con los principios de la presente invención;

La figura 3 ilustra una arquitectura de computadora de ejemplo para estratificar alcance del componente en diferentes niveles de procesamiento de acuerdo con los principios de la presente invención;

La figura 4 ilustra un diagrama de flujo de ejemplo de un método para suministrar aseso de componente de acuerdo con los principios de la presente invención;

La figura 5A 4 ilustra un diagrama de flujo de ejemplo de un método para manejar las actualizaciones de componente de acuerdo con los principios de la presente invención;

La figura 5B 4 ilustra un diagrama de flujo de ejemplo de un método para limitar el alcance de componente de acuerdo con los principios de la presente invención;

6C

66

La figura 6 ilustra un ambiente adecuado para practicar aspectos de la presente invención.

DESCRIPCIÓN DETALLADA DE LAS MODALIDADES PREFERIDAS

La presente invención se extiende a sistemas, métodos, y productos de programa de computador que permiten a los desarrolladores de programa acomodar cambios en los componentes, módulos, y en sistemas operativos sin dañar la función del programa. En particular, se describen sistemas que permiten programas y componentes que acceden uno al otro a través de referencias estáticas o dinámicas para coexistir compatiblemente en un sistema operativo. Las modalidades de la presente invención pueden comprender un propósito una computadora de propósito especial o propósito general incluyendo varios equipos de computador, como se describe con gran detalle adelante.

Las modalidades dentro del alcance de la presente invención también incluyen medios legibles por computadora para llevar o tener instrucciones ejecutables por computadora o estructuras de datos almacenado en estos. Tales medios legibles por computadora pueden ser cualquier medio disponible que pueda ser asesado por un

67

67

computador de propósito general o propósito especial. Por vía de ejemplo, y no de limitación, tales medios legibles por computadora pueden comprender RAM, ROM, EEPROM, CD-ROM u otros dispositivos de almacenamiento de disco óptico, almacenamiento de disco magnético u otros dispositivos de almacenamiento magnético, o cualquier otro medio que pueda ser utilizado para llevar o almacenar medios de códigos de programa deseados en la forma de instrucciones ejecutables por computadora o estructura de datos y que puedan ser accedidos por una computadora de propósito general o una computadora de propósito especial

Cuando la información es transferida o suministrada sobre una red u otra conexión de comunicaciones (sea conectada de manera alámbrica, inalámbrica, o una combinación de comunicación alámbrica o inalámbrica) a una computadora, para computadora propiamente visualiza la conexión como un medio legible por computadora. Así las instrucciones ejecutables por computadora comprenden, por ejemplo, instrucciones y datos que originan una computadora de propósito general, computadora de propósito especial, o dispositivo de procesamiento de propósito especial que desarrolla una cierta función o grupo de funciones.

68

68

La figura 1 muestra una arquitectura de computadora, de ejemplo para practicar una implementación de la presente invención en la cual en módulo determinante 100 recibe una o más solicitudes de un componente solicitante 105 para acceder a otro componente o programa tal como los componentes 120, 125 y 130. Para los propósitos de esta especificación y reivindicaciones, a "un módulo determinante" 100 puede incluir cualquier tipo de módulo de que tiene instrucciones ejecutables configuradas, por ejemplo, para identificar una referencia o un componente objetivo, que incluye el nombre del componente y la versión originalmente pretendida y escoger un componente objetivo apropiado que será utilizado para satisfacer la solicitud. En algunas situaciones, esto puede incluir decidir que no hay tal componente objetivo disponible, de tal forma que el módulo determinante 100 se configuraría para regresar un error. Como se explicará adelante en detalle el módulo determinante 100 puede ser también configurado para identificar otra información tal como la que otros componentes han hecho disponibles para acceso desde un sistema de computadora, proceso, o subproceso.

69

Además, "un componente", para los procesos de esta especificación y reivindicaciones será entendido para intuir cualquier forma de instrucciones ejecutables que puedan ser ejecutadas en un sistema computarizado, como por ejemplo, un archivo de formato de texto interpretado así como también un archivo que ha sido compilado en instrucciones legibles por máquina. El término componente, por lo tanto, puede incluir tanto como programas de aplicaciones grandes y sistemas que suministran una variedad grande de funciones, así como también programas más pequeños y/o componentes de sistema que suministran otros componentes o programas con funcionalidad especifica. Adicionalmente, aunque algunas distinciones serán hechas algunas veces en esta especificación "aplicaciones", "programas de aplicación", "programas", y componentes, las distinciones son simplemente unas de las conveniencias con el fin de clarificar, usualmente, que un conjunto de instrucciones ejecutables son solicitadas o están recibiendo una solicitud para acceder a otro componente en razón de que cada uno puede ser adecuadamente denominado como un componente. De esta manera, los términos "componentes solicitantes" y "componente objetivo" pueden incluir

70

cualquiera de las instrucciones ejecutables anteriores, como se describirá en detalle aquí.

Las modalidades de la presente invención pueden acceder componentes que han sido clasificados "plataformas" o "librería". Los componentes "plataformas" son componentes que pueden ser accedidos por otros múltiples componentes o programas en un sistema computarizado. Los componentes plataforma son normalmente accedidos solamente en la forma más reciente o la forma actualizada de tal forma que el componente solicitante puede simplemente solicitar el componente objetivo generalmente, o una versión mínima del componente objetivo, en lugar de solicitar una versión especifica del componente objetivo. Así, el módulo determinante puede, por ejemplo, ser configurado para suministrar una versión de un componente plataforma diferente de la versión más reciente. En teoría en los componentes plataforma pueden ser sobrescritos mediante un componente actualizado cuando el actualizado es recibido, aunque existan razones por las que no se hace en la práctica. Los componentes plataforma también pueden algunas veces ser denominados como componentes "compatibles binarios". Como contraste, los componentes librería son accedidos por otro componente de programa

21

71

solo si precisamente la misma versión del componente librería ha sido referenciada.

Como se describió en la figura 1, un módulo determinante 100 puede recibir una solicitud o declaración de un componente solicitante 105, para acceder a un componente objetivo, tal como el componente 120, 125, 130. Para el propósito de esta descripción y reivindicaciones, término "componente objetivo" se refiere a un componente para el cual se busca acceso por un componente solicitante. Uno apreciará, sin embargo, que si un componente es un componente objetivo o un componente solicitante es principalmente uno de perspectiva, dependiendo de cual componente solicita acceso del otro. Como tal, la discusión como se aplican los componentes objetivos en esta especificación y reivindicaciones puede aplicar igualmente a los componentes solicitantes, y viceversa.

En cualquier caso, un componente solicitante 105 puede iniciar una solicitud de acceso de componente 110 a través del módulo determinante 100, donde la solicitud indica que el componente solicitante 105 está configurado para acceder a una versión dada del componente objetivo 120, 125 y 130 en algunas implementaciones, la solicitud

72

110, puede ser la referencia encontrada en el código fuente del componente solicitante 105 cuando el componente solicitante 105 es instalado primero en un sistema computarizado dado (no mostrado). Alternativamente, una solicitud 110 puede ser hecha mediante el componente solicitante 105 cuando el componente solicitante 105 solicita acceso de una versión dada de un componente objetivo, tal como los componentes 120, 125, y 130, al momento de correr.

Una "políticas de versiones" para los propósitos de esta especificación y reivindicaciones incluyen cualquiera de un grupo dado de propiedades que pueden ser transportados desde un componente objetivo (por ejemplo 120, 125, 130) a un módulo determinante 100. La política de versiones 131, 132, o 133 especifica si el componente objetivo correspondiente 120, 125, 130 se puede utilizar en lugar de una versión del componente objetivo dado con un número de versión inferior. La política de versiones puede incluir información adicional destinada a ser utilizada por el módulo determinante 100 para decidir si el componente objetivo se puede utilizar en una configuración dada. Así, la política de versiones 132 puede especificar que el componente objetivo 125 (Versión

73

73

1.2.) se puede utilizar cuando la versión 1.1 es solicitada. En algunas modalidades, la política de versiones será encontrada en un sitio predefinido dentro del componente objetivo. En otras implementaciones, la política de versiones se puede transportar al módulo determinante 100 cuando el componente es instalado del sistema, o cuando la primera solicitud es hecha para acceder a un componente dado, y así sucesivamente.

De acuerdo con esto, un componente solicitante puede solicitar acceso a un componente objetivo al solicitar una versión específica del componente objetivo, por ejemplo, solicitud 110 para el "componente 1" "Versión 1.1". Si el componente solicitante 105 solicita, o es configurado para trabajar con una versión específica de un componente objetivo, el módulo determinante 100 puede suministrar el componente solicitante 105 con acceso a una versión específica del componente, dependiendo de la política de versiones 131, 132, 133 que está presente en el componente objetivo. Como se muestra en la figura 1, por ejemplo, el componente solicitante 105 solicita la "versión 1" del "componente 1" al enviar una solicitud 110. De esta manera, el módulo determinante 100 otorga al componente solicitante 105 acceso a la "versión 1.1." del

24

74

"componente 1" aunque una versión más reciente del "componente 1", por ejemplo, "versión 1.2." 125, exista en el sistema.

Como contraste, de algunas modalidades, una solicitud para una versión de un componente, da como resultado acceso a otra versión (por ejemplo actualizado más reciente) de un componente. Por ejemplo, la solicitud 100 puede ser una solicitud para acceder a la "versión 1.1." del "componente 1". Sin embargo, la política de versiones 131 puede indicar que la "versión 1.1." es un componente plataforma (y así una versión más reciente del "componente 1" es suministrada en respuesta a una solicitud). Adicionalmente, en algunas implementaciones podría haber sin embargo múltiples versiones de un componente de plataforma dado en un sistema.

Como tal, el componente solicitante también puede incluir información en esta solicitud que indica la versión más baja posible del componente objetivo plataforma que puede aceptar el componente solicitante 105. Por ejemplo, puede ser que el componente solicitante 105 solicite la "versión 1.4" del componente 1 y que la versión inferior del "componente 1" no sea regresada en respuesta a la

75

solicitud. De acuerdo con esto, el módulo determinante 100 puede suministrar el componente solicitante 105 con acceso a la "versión 3" del "componente 1", aunque la "versión 1.1" y la "versión 1.2." sean accesibles.

Cuando todas las versiones accesibles de un componente solicitado tengan designaciones de versión inferiores a la versión solicitada específica, el módulo determinante 100 puede regresar una respuesta apropiada (por ejemplo un mensaje de error) a un componente solicitante. Por ejemplo, cuando el módulo determinante 100 no tenga acceso a la "versión 3" del "componente 1", al módulo determinante 100 puede enviar un mensaje de error al componente solicitante 105 en respuesta a una solicitud para la "versión 1.4 o mayor" del componente 1".

Puede ser que un número de versión incluya dos partes, una versión y un servicio. Los componentes que tengan un número de versión que indiquen un servicio actualizado se les permiten reemplazar componentes que tengan número de versión que indiquen servicios más viejos. Utilizando manuales de servicio para facilitar el reemplazo del componente es particularmente ventajoso para implementar los cambios menores que tengan una probabilidad reducida

de originar incompatibilidad con otros componentes, para fijar errores, o para fijar problemas de seguridad sean relacionados con los componentes de la librería o plataforma. Esto es, los valores de servicio pueden facilitar "parchear" una versión de un componente. Por ejemplo, si el componente objetivo 120 se identifica como un componente de librería (tal que la versión 1.1 del componente objetivo no sea reemplazada), un desarrollador puede aún actualizar el componente objetivo 120 al actualizar (por ejemplo implementando) un valor de servicio en el número de versión del componente. De acuerdo con esto, el componente objetivo actualizado 120 sería esencialmente un servicio diferente de la "versión 1.1".

La figura 2A ilustra una arquitectura de computadora de ejemplo que recibe versiones más nuevas de los componentes existentes. Esto es, el módulo de determinación 100 puede recibir actualizaciones a los componentes objetivos que ya sean residentes en un sistema computarizado correspondiente. Por ejemplo, el módulo determinante 100 puede recibir los componentes 215 y 210 de un proveedor de servicio de red (no mostrado) conectado a la red 240. El módulo determinante 100 puede

77

recibir los componentes 215 y 210 como el resultado de ejecutar un programa de instalación en el sistema computarizado correspondiente o en el proveedor de servicio de red).

Como se describió, los componentes 210 y 215 incluyen información de política de versiones tal como, por ejemplo del componente actualizado 210 es una actualización "versión 3" del "componente 2", y que el componente es un componente de plataforma. También, el componente 215 puede incluir información en la forma de una política de versiones que el componente 215 es componente de librería, o que el componente 215 configurado de otra forma que a los componentes solicitantes se les puede dar acceso a la versión específica representada por el componente 215.

En respuesta para recibir los componentes 210 y 215, el módulo determinante 100 determina si retener la versión anterior (puede ser denominada como actualización "lado a lado") o reemplazar versiones anteriores (que puede ser denominada como actualización "en el lugar") de cada componente actualizado recibido. Por ejemplo, como se muestra en la figura 2 a los componentes 220 y 235 son

78

78

componentes de librería y plataforma respectivamente. Más específicamente, en respuesta a recibir los componentes 215 el módulo determinante 100 puede identificar que, en razón a que "el componente 1" es un componente de librería, otros programas o componentes sede pueden configurar para acceder específicamente "versión 1" del "componente 1". De acuerdo con esto, el módulo determinante 100 puede determinar que ambos componentes 215 y el componente 220 deban ser retenidos.

Más específicamente en respuesta a recibir el componente 210, el módulo determinante 100 puede identificar que, la zona a que "el componente 2" es un componente de plataforma, los programas y componentes solicitantes les será dado acceso a la versión más reciente del componente 2. De acuerdo con esto, el módulo determinante 100 puede determinar que el componente 235 va hacer reemplazado con el componente 210.

La figura 2B ilustra por ejemplo la arquitectura de computadora de la figura 2 A después de que un módulo de determinación ha determinado las versiones de los componentes que van a ser retenidas. Como se describió en la figura 2B la "versión 1" (componente 220) cómo la

79

"versión 2" (componente 215) del componente 1 permanece en el sistema (una actualización lado a lado). También como se describió en la figura 2B, solamente la "versión 3" del "componente 2" (componente 210) permanece en el sistema (una actualización en el lugar).

La figura 3 ilustra la arquitectura de computadora de ejemplo para estratificar el alcance del componente en diferentes niveles de procesamiento de acuerdo con una implementación de la presente invención. La estratificación se basa en el alcance del componente que aplica a los componentes objetivo. Por vía de explicación y no de limitación la figura 3 indica 3 niveles de alcance, es decir, un nivel de "Máquinas" 330, un nivel de "Proceso" 340, y un nivel de "Sub-Procesos" 350. Uno apreciará como sin embargo, después de leer esta descripción y reivindicaciones que podría haber más o menos números de niveles, según sea apropiado. En particular, los aspectos de la invención le permiten al componente objetivo suministrar una política de versiones que requiera que solamente una versión del objetivo sea hecha disponible a un nivel dado pero entonces es decir (solo una versión en la máquina completa o solamente una

80

en un Proceso dado o solamente una en un Sub-Proceso dado).

Por ejemplo, una política de versiones que está asociada con un componente objetivo dado 300 puede incluir un conjunto de alcance de componente. En relación brevemente de nuevo con la figura 1 el alcance del componente puede indicar que un componente solicitante 105 debe acceder al componente objetivo 300 a un nivel de proceso dado como se muestra en la figura 3, por ejemplo, "el componente 1" "Versión 1" 300 se identifica para acceso a nivel de máquina. Cualquier componente solicitante instalado en el sistema que solicite acceso del componente objetivo 300, debe utilizar el "componente 1" "versión 1" en razón a que el componente objetivo 300 se configura para acceso a nivel de máquina. Como con otras propiedades de política de versiones, esta limitación del nivel de proceso puede ser indicada por el desarrollador de un componente objetivo antes de que el componente sea instalado en un sistema dado.

El alcance del componente también puede indicar mayor o menor alcance para un componente objetivo 300, 310, 315, 320 y 325. Por ejemplo, las "política de versiones"

identificadas con un componente dado 310 pueden indicar que una versión dada del componente objetivo 310 sea requerida solamente dentro de un proceso cierto 342, 345, o Sub-Proceso 352, 355. Como se mostró en la figura 3, por ejemplo, cualquier componente solicitante 105 que solicite acceso a una versión dada de un componente 310 puede hacerlo así dentro de un proceso 342 sin requerir que otros componentes solicitantes (en el sistema) utilicen el nuevo componente objetivo en otros procesos 315. Como tal, el componente 310 puede ser utilizado en el proceso 342, aunque el componente 315 puede ser utilizado en el proceso 345. Adicionalmente, cuando el proceso A 342 lo ha seleccionado una versión particular el Sub-Proceso 350 como que depende del proceso 340, puede utilizar diferentes versiones de los componentes 310, tal como los componentes 320 y 325. Este nivel de acceso granular a diferentes componentes puede ser por lo tanto indicado cuando un componente dado es desarrollado, en lugar de mediante un administrador de sistema cuando el componente dado se instala en el sistema. El módulo determinante 100 puede combinar cualquier alcance de componente identificado para cada componente objetivo o solicitante para suministrar un componente solicitante con acceso de componente objetivo apropiado.

82

De acuerdo con esto la identificación de una versión apropiada de un componente objetivo puede ser basada en otras políticas, como tal, por ejemplo, el componente. Un módulo determinante puede por lo tanto identificar una versión apropiada de un componente objetivo con base en cualquier política identifica, tal como la política de versiones y el alcance de componente del componente de objetivo especificado, así como también cualquiera otra política suministradas por el administrador del sistema cuando sea apropiado.

La presente invención puede también ser descrita en términos de métodos que comprenden etapas funcionales y/o actos no funcionales. Las figuras 4, 5A y 5B ilustran diagramas de flujo de ejemplo para permitirle acceso al componente por otros programas o componentes en un sistema computarizado. Los métodos de las figuras 4, 5A y 5B serán discutidos con respecto a los módulos de los programas ilustrados en las figuras precedentes.

La figura 4 ilustra un diagrama de flujo de ejemplo de un método para suministrar acceso de componente de acuerdo con una implementación de la presente invención. El

83

83

método de la figura 4 incluye un acto 400 de recibir una solicitud para una versión de un componente objetivo. El acto 400 puede incluir solicitar una solicitud para acceder a una versión especificada de un componente objetivo, la solicitud siendo recibida de un componente solicitante. Por ejemplo, un componente solicitante 105 puede solicitar acceso de un componente objetivo, tal como el componente 120, 125, y 130 a través de un módulo determinante 100. Podría ser también que la solicitud incluya la política de versiones de la versión del componente objetivo.

El método tan bien incluye una etapa orientada por su estado funcional 440 de suministrar un componente objetivo apropiado. La etapa 440 puede incluir cualquier número de actos correspondientes para implementar la presente invención. Sin embargo, como se describió en la figura 4 la etapa 440 incluye un acto 410 de identificar una política de una versión. El acto 410 puede incluir e identificar una política de versión de la versión especificada del componente objetivo. Si una política de versión fue incluida en la solicitud el módulo de determinar si el puede identificar tal política de versión incluida. Alternativamente el módulo de

91

84

determinar 100 puede referirse a una o más versiones del componente objetivo e identificar las políticas de versión almacenada en un a o más versiones del componente objetivo. Por ejemplo, el módulo determinante si el puede identificar que versiones múltiples de un componente tales como "versión 1" 120 y "versión 2" 125 del mismo "componente 1" existen en el sistema, teniendo cada una política de versiones correspondiente 131, 132 y 133. Un desarrollador de software puede incluir una política de versión en componentes objetivos 120, 125 y 130 etc., de tal forma que el módulo determinante 100 identifica la política de versiones al compilar, instalar, y/o correr el programa desarrollado

La etapa 440 también incluye un acto 430 de suministrar una versión propiedad del componente objetivo. El acto 430 puede incluir identificar una versión apropiada del componente objetivo basado en la política mencionada del componente objetivo especificado. Por ejemplo el módulo determinante 100 puede suministrar un componente solicitante 105 con una versión específica del componente objetivo solicitado (un componente de librería) tal como el componente 120. Alternativamente, el módulo determinante 100 puede suministrar el componente

solicitante 105 con una versión más reciente de un componente (un componente de plataforma), tal como el componente 130.

La figura 5A ilustra un diagrama de flujo ejemplo de un método para manejar actualizaciones de componentes de acuerdo con una implementación de la presente invención. El método de la figura 5A puede ser implementado de modo que un componente de solicitud que accesa al componente objetivo continúa operando efectivamente después de que el componente objetivo ha sido actualizado. Como se ilustró, el método ilustrado en la figura 5A incluye un acto 500 para recibir una actualización de un componente. El acto 500 puede también incluir identificar que un componente objetivo es accesado por un componente de solicitud. Por ejemplo, haciendo referencia de nuevo a la figura 1, un módulo de determinación 100 puede estar ligado a, o contener, un registro o una base de datos que, luego de la instalación de un componente de solicitud 105, identifica que el programa instalado o el componente o componente 105 está configurado para accesar una versión específica de un componente objetivo tal como una "versión 1" 120 del "componente 1". Ese módulo de determinación 100 puede ganar esta información con base

en cualquier política de versiones contenidas en el programa instalado, lo mismo que contenidas dentro de cualesquier componentes que el programa está configurado para accesar.

El método ilustrado en la figura 5A incluye un acto 510 de identificación de una política de versiones. El acto 510 puede incluir identificar una política de versiones en una versión anterior del componente objetivo, y una versión actualizada del componente objetivo. Por ejemplo, el módulo de determinación 100 identifica la política de versiones en cualquiera de los componentes objetivos 120, 125, y 130, que, como se describió previamente, puede indicar la versión del componente objetivo 120, 125, y 130, y puede indicar que el componente objetivo tiene el propósito de ser un componente de plataforma o de librería.

El método ilustrado en la figura 5A incluye adicionalmente un acto 520 de agregar la actualización de componente al sistema con base en por lo menos en parte sobre la política de versiones. El acto 520 puede incluir eliminar una versión anterior del componente objetivo y/o agregar la versión actualizada del componente objetivo

87

con base en la política de versiones identificada. Por ejemplo, en una versión anterior específica 220 de un componente es requerida para el acceso por otro programa o componente de modo que si el componente es un componente de librería, el módulo de determinación 100 no sobrescribirá la versión anterior 220. El módulo de determinación 100 simplemente agregará la nueva versión 215 del componente de modo que los programas o componentes que requieran una nueva versión 215 del componente lo pueden accesar. Similarmente, los programas o componentes que requieran la versión anterior 220 del componente también la pueden accesar igualmente, lo que preserva la integridad del programa de solicitud o componente. En contraste, si ningún programa o componente es identificado en cuanto a que requiera una versión específica de un componente dado (un componente de plataforma), el módulo de determinación 100 puede simple sobrescribir la versión anterior 235 del componente con la versión reciente 235 del componente con la versión reciente 210 del componente.

La figura 5B ilustra un diagrama de flujo ejemplo de un método para suministrar acceso al componente en uno o más niveles de proceso de acuerdo con una implementación de

88

la presente invención. El método de la figura 5B puede ser implementado para organizar uno o más componentes objetivo de modo que se tenga acceso a uno o más componentes objetivo de manera limitada. El método ilustrado en la figura 5B incluye un acto 550 para identificación de una política de versiones. El acto 550 puede incluir identificar una política de versiones en un componente objetivo. Por ejemplo, un módulo de determinación 100 puede recibir una solicitud de acceso a componentes de un componente de solicitud, y puede identificar una versión de entre una actualización recibida 215, 210, dentro de un componente objetivo existente 220, 225, etc. Como se describió previamente, la política de versiones puede ayudar al sistema a identificar un número de versión del componente objetivo, lo mismo que si el componente objetivo 220, 225, y 230 tiene el propósito de ser un componente de librería o de plataforma.

El método ilustrado en la figura 5B incluye un acto 560 para identificar un alcance de componente asociado con el componente. El acto 560 puede incluir identificar un alcance de componente asociado con el componente objetivo en donde el alcance de componente identifica una

89

propiedad que está asociada con un componente de solicitud que puede ser configurado para accesar el componente objetivo. Por ejemplo, sea que el componente objetivo o el programa o componente de solicitud pueda estar asociado con un alcance de componente específico que indique que una versión del componente objetivo puede ser acezada en una de las capàs de la máquina, una capa de subproceso, etc. Uno podrá apreciar, sin embargo, como se describió aquí, que puede haber una variedad de niveles en los cuales el acceso al componente pueda ser limitado, dependiendo de las preferencias del desarrollador.

El método ilustrado en la figura 5B incluye un acto 570 de permitir el acceso al componente objetivo con base en el alcance del componente y la política de versiones. El acto 570 puede incluir permitir que por lo menos uno entre uno o más componentes de solicitud tenga acceso al componente objetivo con base en la propiedad de acceso asociada con un componente de solicitud y la política e versiones identificada. Por ejemplo, si uno o más programas o componentes 300 son indicados por procesos de máquina amplia 330, solamente aquella versión del componente 300 estará disponible a cualquiera de los

90

componentes de solicitud en un nivel de proceso dado. En contraste, si el componente objetivo es identificado con el acceso de nivel de proceso, el módulo de determinación 100 puede permitir que otros componentes de solicitud tengan acceso a diferentes versiones del mismo componente objetivo para un proceso correspondiente dado o subproceso dado, según sea apropiado.

La figura 6 y la siguiente discusión tienen el propósito de suministrar una descripción breve general del ambiente de cómputo adecuado en el cual la invención puede ser implementada. Aunque no es requerido, la invención será descrita en el contexto general de instrucciones ejecutables por computador, tal como módulos de programa, que son ejecutados por computadoras en ambientes de red. Generalmente, los módulos de programas incluyen rutinas, programas, objetos, componentes, estructuras de datos, etc., que llevan a cabo tareas particulares o implementan tipos de datos abstractos particulares. Las instrucciones ejecutables por computadora, asociados a estructuras de datos, y los módulos de programa representan ejemplos de los medios de codificación de programa para ejecutar las etapas de los métodos descritos aquí. La secuencia particular de tales instrucciones ejecutables o

91

estructuras de datos asociados representa ejemplos de actos correspondientes para implementar las funciones descritas en tales etapas.

Aquellos versados en la técnica podrán apreciar que la invención puede ser practicada en ambientes de cómputo en red con muchos tipos de configuraciones de sistemas de computadoras, incluyendo computadoras personales, dispositivos de mano, sistemas de multiprocesadores, equipos electrónicos de consumo programables o con base en microprocesadores, PC de red, mini computadores, computadoras medianas y similares. La invención también puede ser practicada en ambientes de cómputo distribuidos en donde las tareas se llevan a cabo por dispositivos de procesamiento locales y remotos que están enlazados (sea por enlaces físicos, enlaces inalámbricos, o por una combinación de enlaces físicos o inalámbricos) a través de una red de comunicaciones. En un ambiente de cómputo distribuido, los módulos de programa pueden estar localizados tanto en dispositivos de almacenamiento de memoria locales como remotos.

Aquellos versados en la técnica podrán apreciar que la invención puede ser practicada en ambientes de cómputo en

92

red con diversos tipos de configuraciones de sistemas de computadoras, incluyendo computadoras personales, dispositivos de mano, sistemas de multiprocesadores, aparatos electrónicos de consumo programables o con base en microprocesadores, PC de red, minicomputadoras, computadoras medianas, y similares. La invención también puede ser practicada en ambientes de cómputo distribuidos en donde los dispositivos de procesamiento remoto o locales llevan a cabo tareas y están enlazados (sea por enlaces físicos, enlaces inalámbricos, o por una combinación de enlaces físicos o inalámbricos) a través de una red de comunicaciones. En un ambiente de cómputo distribuido, los módulos de programa pueden estar localizados tanto en dispositivos de almacenamiento de memoria locales como remotos.

Con referencia a la figura 6, un sistema ilustrativo para implementar la invención incluye un dispositivo de cómputo de propósito general en la forma de un computador convencional 620, que incluye una unidad de procesamiento 621, una memoria del sistema 622, y un bus del sistema 623 que acopla los diversos componentes del sistema incluyendo la memoria del sistema 622 a la unidad de procesamiento 621. El bus del sistema 623 puede ser

93

93

cualquiera de diversos tipos de estructura de bus incluyendo un bus de memoria o controlador de memoria, un bus de periféricos y un bus local que usa cualquiera de una variedad de arquitecturas de bus. La memoria del sistema incluye memoria de solo lectura (ROM) 624 y memoria de acceso aleatorio (RAM) 625. Un sistema básico de entrada/salida (BIOS) 626, contiene las rutinas básicas que ayudan a transferir la información entre los elementos dentro del computador 620, tal como durante el arranque, puede estar almacenado en una ROM 624.

El computador 620 puede también incluir un dispositivo de disco duro magnético 627 para leer desde y escribir en un disco duro magnético 639, un dispositivo de disco magnético 628 para leer desde o escribir en un disco magnético removible 629, y un dispositivo de disco óptico 630 para leer desde o escribir a en un disco óptico removible 631 tal como un CD-ROM u otro medio óptico. El dispositivo de disco duro magnético 627, el dispositivo de disco magnético 628, y el dispositivo de disco óptico 630 están conectados al bus del sistema 623 por una interfaz de dispositivo de disco duro 632, una interfaz de dispositivo de disco magnético 633, y un interfaz de dispositivo óptico 634, respectivamente. Los dispositivos

94

94

y sus medios legibles por computador asociados suministran almacenamiento no volátil de las instrucciones ejecutables por computador, las estructuras de datos, los módulos de programa y otros datos para el computador 620. Aunque el ambiente ilustrativo descrito aquí emplea un disco duro magnético 639, un disco magnético removible 629 y un disco óptico removible 631, se pueden usar otros tipos de medios legibles por computadora para almacenar datos, incluyendo cassettes magnéticos, tarjetas de memoria flash, discos versátiles digitales, cartuchos de Bernoulli, RAM, ROM y similares.

Los medios de código de programa comprenden uno o más módulos de programa que pueden ser almacenados en el disco duro 639, disco magnético 629 o disco óptico 631, ROM 624 o RAM 625, incluyendo un sistema operativo 635, uno o más programas de aplicación 636, otros módulos de programa 637, y datos de programa 638. un usuario puede ingresar comandos e información dentro del computador 620 por medio del teclado 640, el dispositivo de señalamiento 642 u otros dispositivos de entrada (no mostrados), tales como un micrófono, un Joystick, una almohadilla de juego, un plato de satélite, un escáner, o similares. Estos y otros dispositivos de entrada pueden estar frecuentemente

95

95

conectados a la unidad de procesamiento 621 por medio de una interfaz de puerto serial 646 acoplado al bus del sistema 623. Alternativamente, los dispositivos de entrada pueden estar conectados por otras inveraces, tales como un puerto paralelo, un puerto de juegos o un bus serial universal (USB). Un monitor 647 u otro dispositivo de visualización también está conectado al bus del sistema 623 por medio de una interfaz, tal como un adaptador de video 648. En adición al monitor, los computadores personales típicamente incluyen otros dispositivos de salida periféricos (no mostrados), tales como parlantes e impresoras.

El computador 620 puede operar en un ambiente de red usando conexiones lógicas a uno o más computadores remotos, tal como computadores remotos 649a y 649b. Los computadores remotos 649a y 649b pueden cada uno ser computadores personales, un servidor, un enrutador, un PC de red, un dispositivo par u otro nodo de red común, y típicamente incluye muchos o todos de los elementos descritos anteriormente con relación al computador 620, aunque solamente dispositivos de almacenamiento de memoria 650a y 650b y sus programas de aplicación asociados 636a y 636b han sido ilustrados en la figura 6.

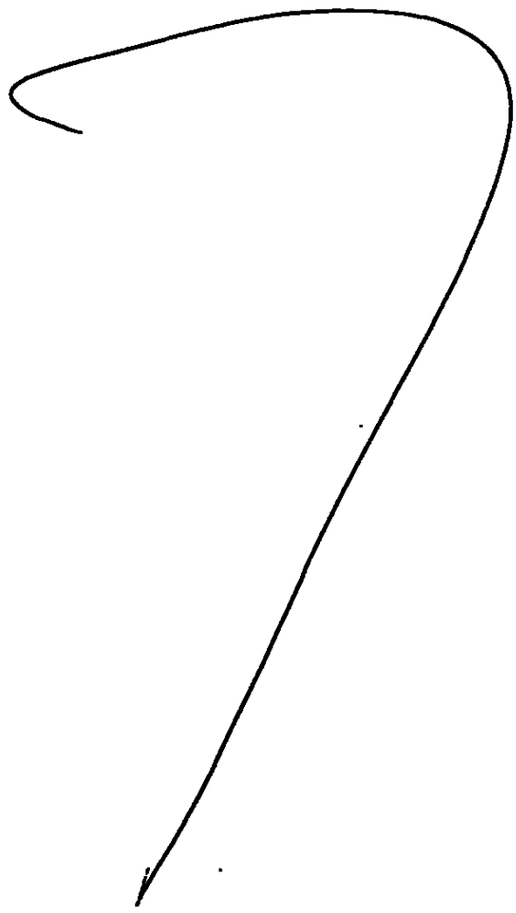
96

Las conexiones lógicas ilustradas en la figura 6 incluyen una red de área local (LAN) 651 y una red de área amplia (WAN) 652 que están presentes aquí como vía de ejemplo y no como limitación. Tales ambientes de red son lugares comunes en oficinas grandes o empresas grandes que tienen redes de computadoras, intranet e Internet.

Cuando se usó en un ambiente de red LAN, el computador 620 está conectado a la red local 651 por medio de una interfaz o adaptador de red 653. Cuando es usado en un ambiente de red WAN, el computador 620 puede incluir un módem 654, un enlace inalámbrico, u otros medios para establecer comunicaciones con una red de área amplia 652, tal como la Internet. El módem 654, que puede ser interno o externo, está conectado al bus del sistema 623 por medio de una interfaz de puerto serial 646. En un ambiente en red, los módulos de programa ilustrados con relación al computador 620, o porciones de ellos pueden estar almacenados en dispositivos de almacenamiento de memoria remotos. Esto podrá ser apreciado en que las conexiones de red mostradas son ilustrativas y se pueden usar otros medios de establecer comunicaciones en una red de área amplia 652.

97

La presente invención puede ser visualizada en otras formas específicas sin salirse de su espíritu y características esenciales. Las modalidades descritas deben considerarse en todos los respectos solamente como ilustrativas y no restrictivas. El alcance de la invención es, por lo tanto, indicado por las reivindicaciones adjuntas en vez de por las descripción anterior. Todos los cambios que caigan dentro del significado y rango de equivalencia de las reivindicaciones deben considerarse dentro de su alcance.



98

98

REIVINDICACIONES

Nosotros reclamamos:

1. En un sistema computarizado que incluye uno o más componentes de programa que incluyen uno o más componentes de solicitud que pueden solicitar acceso a uno o más componentes objetivo, un método para suministrar un componente de solicitud con acceso a una versión apropiada de un componente objetivo, comprende los actos de:

   Recibir una solicitud para accesar una versión especificada de un componente objetivo, la solicitud es recibida desde un componente de solicitud;

   Identificar una política de versiones de la versión especificada del componente objetivo;

   Identificar una versión apropiada del componente objetivo con base en la política de versiones del componente objetivo especificado;

99

Suministrar el componente de solicitud con acceso a la versión apropiada del componente objetivo.

2. El método según la reivindicación 1, en donde la versión solicitada del componente objetivo es una entre un componente de librería y un componente de plataforma.

3. El método según la reivindicación 1, en donde identificar una versión apropiada del componente objetivo comprende identificar una versión más reciente del componente objetivo en respuesta a un solicitud para una versión anterior del objetivo aún cuando la versión más reciente y la versión anterior ambas sean accesibles al sistema computarizado.

4. El método según se da en la reivindicación 3, en donde identificar una versión más reciente del componente objetivo en respuesta a una solicitud de una versión anterior del objetivo aún cuando la versión más reciente y la versión anterior ambas sean accesibles al sistema computarizado comprende identificar una versión más reciente de

100

un componente de plataforma aún cuando una versión anterior del componente de plataforma permanezca en el sistema cuando la versión más reciente fue recibida en el sistema computarizado.

5. El método según la reivindicación 1, en donde la política de versiones de la versión especificada del componente objetivo es identificada cuando el componente objetivo es uno o más entre compilados, configurados, instalados o que corren en el sistema computarizado.

6. El método según la reivindicación 1, en donde la información de la versión que identifica el componente objetivo especificado es almacenada en el componente de solicitud cuando el componente de solicitud es uno o más entre compilados, configurados, instalados o que corren en el sistema computarizado.

7. El método según la reivindicación 1, que comprende adicionalmente:

101

identificar uno o más componentes de solicitud que sean capaces de accesar una versión anterior del componente objetivo;

identificar que ninguno de los uno o más componentes de solicitud estén configurados para accesar la versión anterior del componente objetivo; y

eliminar la versión anterior del componente objetivo.

8. El método según la reivindicación 1, en donde la versión apropiada del componente objetivo es la versión del componente objetivo que fue solicitada.

9. El método según la reivindicación 1, en donde la versión apropiada del componente objetivo es diferente de la versión del componente objetivo que fue solicitado.

10. El método de la reivindicación 9, en donde el acceso al componente objetivo es suministrado al

192

102

componente de solicitud por medio de un módulo de determinación.

11. El método según la reivindicación 10, en donde la disponibilidad de una o más de las versiones anteriores del componente objetivo y las más versiones recientes del componente objetivo son identificadas por un módulo de determinación cuando la una o más de las versiones anteriores del componente objetivo y las más de las recientes versiones del componente objetivo son recibidas por el sistema computarizado.

12. El método según la reivindicación 1, en donde la política de versiones es insertada en instrucciones ejecutables por computadora en el componente objetivo antes de uno instalado, configurado, o ejecutado de los componentes objetivos en el sistema computarizado.

13. El método según la reivindicación 1, en donde la política de versiones es identificada adicionalmente como cualquier versión del componente objetivo.

193

103

14. El método de la reivindicación 13, en donde la política de versiones identifica que cualquiera de las versiones anteriores del componente objetivo y la versión más reciente del componente objetivo se configuran para ser accesados por una versión específica del componente de solicitud.

15. El método según la reivindicación 1, que comprende adicionalmente identificar un alcance de componente que está asociado con el componente objetivo.

16. El método según la reivindicación 15, en donde el acceso a una versión especificada del componente objetivo está basado adicionalmente en uno de los alcances de componente identificado asociado con el componente objetivo, y un alcance de componente objetivo suministrado por un administrador del sistema.

17. El método según la reivindicación 16, en donde el alcance de componente identificado especifica que el acceso a la versión especificada del

componente objetivo es suministrada en uno o más de los niveles de máquina, nivel de proceso o nivel de subproceso.

18. El método según la reivindicación 1, en donde el componente objetivo solicitado es un componente de librería, el método comprende adicionalmente identificar un valor de servicio asociado con el componente objetivo solicitado.

19. El método según la reivindicación 1, en donde identificar una versión apropiada del componente objetivo comprende identificar una versión actualizada de un componente de librería con base en la política de versiones identificada y el valor de servicio identificado.

20. En un sistema computarizado que incluye uno o más componentes de programa que incluyen uno o más componentes de solicitud que pueden solicitar acceso a uno o más componentes objetivo, un método para suministrar un componente de solicitud con acceso a una versión apropiada de un componente objetivo, comprende:

105

un acto recibir una solicitud para accesar una versión especificada de un componente objetivo, la solicitud es recibida desde un componente de solicitud;

una etapa para permitir el acceso a una versión apropiada del componente objetivo solicitado de modo que el componente de solicitud accesa el componente objetivo apropiado a medida que este ha sido configurado para hacerlo así, y de modo que el componente de solicitud no falle cuando solicite acceso a un componente que ha sido actualizado.

21. El método según la reivindicación 20, en donde la etapa para permitir el acceso a una versión apropiada del componente objetivo solicitado comprende los actos correspondientes de:

Identificar una política de versiones de la versión especificada del componente objetivo;

196

106

Recibir una solicitud para accesar una versión especificada de un componente objetivo, la solicitud empieza siendo recibida desde un componente de solicitud;

Suministrar el componente de solicitud con acceso a la versión apropiada del componente objetivo.

22. En un sistema computarizado que incluye uno o más componentes de programa que incluyen uno o más componentes de solicitud que pueden solicitar acceso a uno o más componentes de solicitud que pueden solicitar acceso a uno o más componentes objetivos, un método para actualizar un componente objetivo de modo que un componente de solicitud que accese al componente objetivo continúe operando efectivamente después de que el componente objetivo ha sido actualizado, comprende los actos de:

Identificar que un componente de solicitud está configurado para accesar un componente objetivo;

107

107

Identificar una polít. a de versiones en por lo menos una versión existente de un componente objetivo y una versión previamente instalada del componente objetivo; e

Identificar qué versiones del componente objetivo deben permanecer en el sistema con base en cualquier política de versiones identificada que corresponda a por lo menos la versión existente del componente objetivo y la versión previamente instalada del componente objetivo.

23. El método según la reivindicación 22, que comprende adicionalmente recibir una versión actualizada del componente objetivo a través de una red desde un proveedor de servicios de red.

24. El método según la reivindicación 22, en donde, si la política de versiones indica que el componente de solicitud es un componente de librería, agregar la versión existente del componente objetivo al sistema sin remover la versión previamente instalada del componente objetivo.

108

25. El método según la reivindicación 22, en donde, si la política de versión indica que el componente de solicitud es un componente de plataforma, sobrescribir la versión instalada previamente del componente objetivo con la versión existente del componente objetivo.

26. En un sistema computarizado que incluye uno o más componentes de solicitud que son configurados para accesar uno o más componentes fuente, un producto de programa de computador que tiene instrucciones ejecutables por computador almacenado allí que, cuando son ejecutadas, causan que el sistema computarizado lleve a cabo un método para suministrar un componente de solicitud con acceso a una versión apropiada de un componente objetivo, comprende los actos de:

Recibir una solicitud para accesar una versión especificada de un componente objetivo, la solicitud es recibida desde un componente de solicitud;

109

Identificar una política de versiones de la versión especificada del componente objetivo;

Identificar una versión apropiada del componente objetivo con base en la política de versiones del componente objetivo especificado;

Suministrar el componente de solicitud con acceso a la versión apropiada del componente objetivo.

27. En un sistema computarizado que incluye uno o más componentes de solicitud que son configurados para accesar uno o más componentes fuente, un producto de programa de computadora que tiene instrucciones ejecutables por computadora almacenados en ella que, que cuando son ejecutados, causan que el sistema computarizado lleve a cabo un método para actualizar un componente objetivo de modo que un componente de solicitud que accese un componente objetivo continúe operando efectivamente después de que el componente objetivo ha sido actualizado, comprende los actos de:

110

110

Identificar que un componente de solicitud es configurado par accesar a un componente objetivo;

Identificar una política de versiones en por lo menos una versión existente del componente objetivo y una versión instalada previamente del componente objetivo; e

Identificar qué versiones del componente objetivo deberían permanecer en el sistema con base en cualquier política de versiones identificada correspondiente a por lo menos la versión existente del componente objetivo y la versión previamente instalada del componente objetivo.

RESUMEN

Una política de versiones incluida en un componente objetivo indica cómo el componente objetivo es accesado, por ejemplo, sea como un componente de librería o como un componente de plataforma. Un componente puede ser designado como un componente de librería cuando su versión no está conformada de una manera compatible y binaria. Cuando otros componentes solicitan tal componente ellos reciben específicamente la versión del componente que ellos han solicitado. Por otro lado, un componente puede ser designado como un componente de plataforma cuando su versión se ha logrado de una manera compatible y binariamente. Cuando otros componentes solicitan tal componente ellos pueden recibir la última versión actualizada del componente solicitado en su lugar. Así, se suministra el acceso a una versión apropiada del componente (aún cuando una versión difiera de la versión solicitada). Otras modalidades incluyen mecanismos para estratificar el alcance del componente con base en niveles de procesamiento diferentes.

Título: SOPORTE DE VERSIONES EN LENGUAJES Y HERRAMIENTAS DE PROGRAMACIÓN ORIENTADAS A OBJETO
Inventores: James S. Miller, et al.
Registro No. 768.493

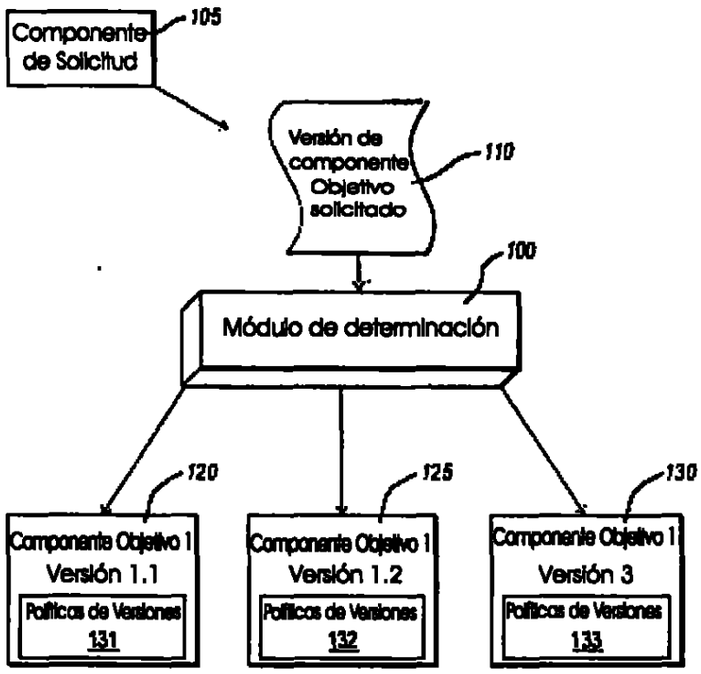


Fig. 1

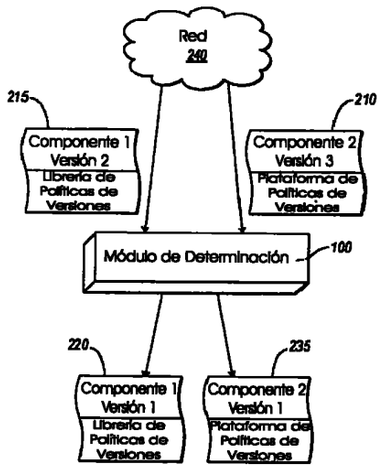

Red
240
215
Componente 1
Versión 2
Librería de
Políticas de
Versiones
210
Componente 2
Versión 3
Plataforma de
Políticas de
Versiones
Módulo de Determinación
100
220
Componente 1
Versión 1
Librería de
Políticas de
Versiones
235
Componente 2
Versión 1
Plataforma de
Políticas de
Versiones

Fig. 2A

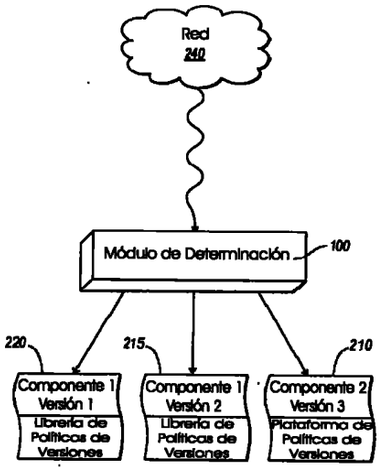

Red
240
Módulo de Determinación
100
220
Componente 1
Versión 1
Librería de
Políticas de
Versiones
215
Componente 1
Versión 2
Librería de
Políticas de
Versiones
210
Componente 2
Versión 3
Plataforma de
Políticas de
Versiones

Fig. 2B

Título: SOPORTE DE VERSIONES EN LENGUAJES Y HERRAMIENTAS DE PROGRAMACIÓN ORIENTADAS A OBJETO
Inventores: James S. Miller, et al.
Registro No. 13768.493

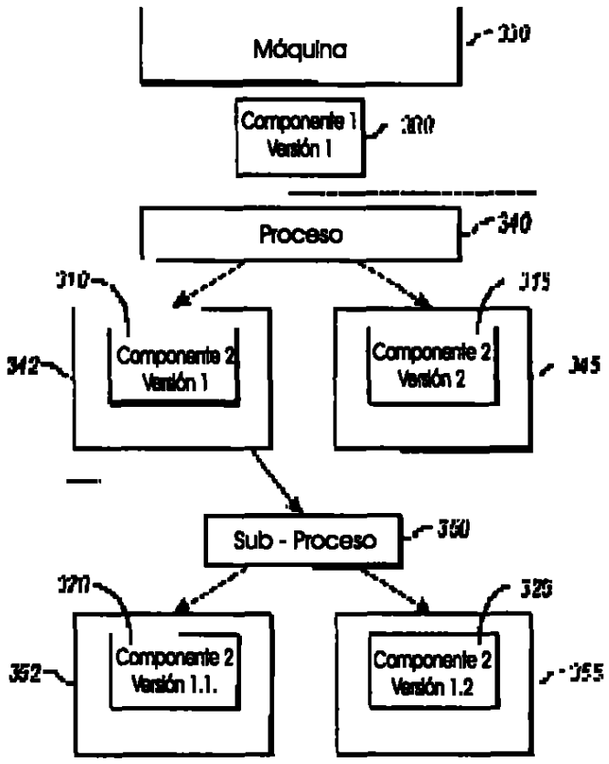


Fig. 3

Título: SOPORTE DE VERSIONES EN LENGUAJES Y HERRAMIENTAS DE PROGRAMACIÓN ORIENTADAS A OBJETO
Inventores: James S. Miller, et al.
Registro No. 13768.493

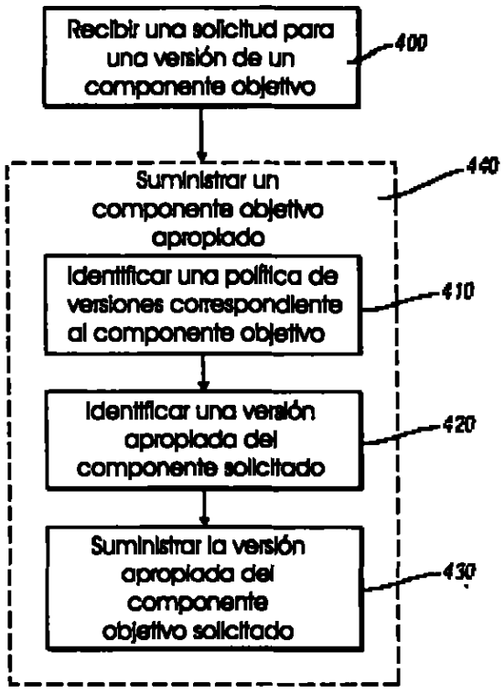


Fig. 4

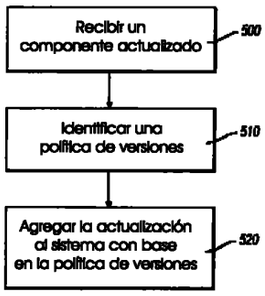

Recibir un componente actualizado
500
Identificar una política de versiones
510
Agregar la actualización al sistema con base en la política de versiones
520

Fig. 5A

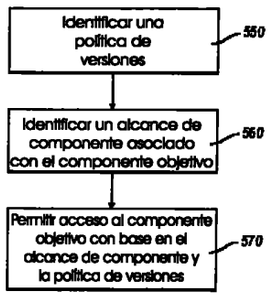

Identificar una política de versiones
550
Identificar un alcance de componente asociado con el componente objetivo
560
Permitir acceso al componente objetivo con base en el alcance de componente y la política de versiones
570

Fig. 5B

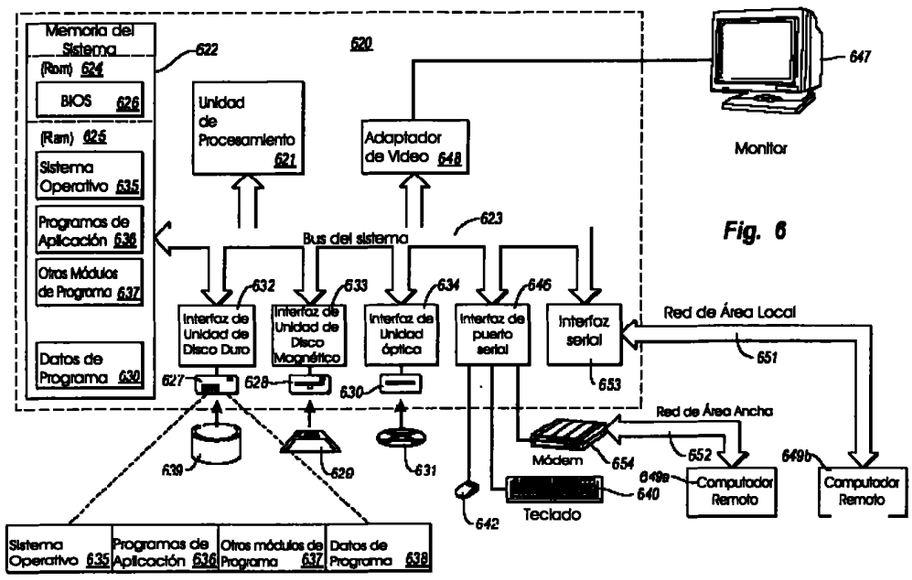

Memoria del Sistema
(Rom) 624
622
620
BIOS 626
(Ram) 625
Sistema Operativo 635
Programas de Aplicación 636
Otros Módulos de Programa 637
Datos de Programa 638
Unidad de Procesamiento 621
Adaptador de Video 648
647
Monitor
623
Bus del sistema
632
633
634
646
Interfaz de Unidad de Disco Duro
Interfaz de Unidad de Disco Magnético
Interfaz de Unidad óptica
Interfaz de puerto serial
Interfaz serial
653
Red de Área Local
651
627
628
630
639
629
631
Red de Área Ancha
Módem
654
652
649a
640
Computador Remoto
649b
Computador Remoto
642
Teclado
Sistema Operativo 635
Programas de Aplicación 636
Otros módulos de Programa 637
Datos de Programa 638

Fig. 6

119
118

SUPERINDUSTRIA Y COMERCIO
Radicacion : 04128509 00000000 Folios: 118
Fecha (AMD): 2004-12-24 18:26:26
Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

REQUISITOS NECESARIOS PARA PRESENTACION Y AGILIZACION DEL TRAMITE

	USUARIO	RADICADOR
PATENTE DE INVENCIÓN / *MODELO DE UTILIDAD*	( )	(✓)
- 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 gràfica del esquema de trazado	( )	( )
Comprobante de pago de las tasas establecidas.	( )	( )
COMPLETA ( ) INCOMPLETA	( )	( )

Firma Responsable:

Fecha:

PROTOCOLO

Expediente 04-128509

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

120

Industria y Comercio
SUPERINTENDENCIA

Folio 121

2020
Bogotá, D. C.

Doctor(a)
DARIO CARDENAS NAVAS
Apoderado(a)

Oficio No. 12068

REFERENCIA:	**Radicación**	**04 128509**
	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 el inventor 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 Única No.10 de 2001.

ALIX CARMENZA CÉSPEDES DE VERGEL
Jefe de la División de Nuevas Creaciones

El anterior requerimiento fue notificado por fijación en lista de fecha 10 FEB. 2005

Se recuerda al interesado que debe presentar, dentro de los dieciséis meses siguientes contados a partir de la fecha de presentación de la solicitud cuya prioridad se invoca, copia de la misma certificada por la autoridad que la expidió y, en los casos en que haya lugar, la traducción simple del capítulo reivindicatorio. El incumplimiento de éste plazo, acarreará la pérdida de la prioridad invocada.

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