PCT ORGANISATION MONDIALE DE LA PROPRIETE INTELLECTUELLE
Bureau international

DEMANDE INTERNATIONALE PUBLIEE EN VERTU DU TRAITE DE COOPERATION EN MATIERE DE BREVETS (PCT)

(51) Classification internationale des brevets 6 :		(11) Numéro de publication internationale:
G06F 9/44	A1	WO 99/35566
		(43) Date de publication internationale: 15 juillet 1999 (15.07.99)

(21) Numéro de la demande internationale: PCT/FR98/02887

(22) Date de dépôt international: 28 décembre 1998 (28.12.98)

(30) Données relatives à la priorité:
97/16700 30 décembre 1997 (30.12.97) FR

(71) Déposant *(pour tous les Etats désignés sauf US)*: BULL S.A. [FR/FR]; 68, route de Versailles, F–78430 Louveciennes (FR).

(72) Inventeurs; et
(75) Inventeurs/Déposants *(US seulement)*: CHAUVALON, Denis [FR/FR]; 34, rue Aristide Briand, F–92170 Vanves (FR). CHEVAL, Bernard [FR/FR]; 12, square de l'Arno, F–78180 Montigy le Bretonneux (FR). DE MONES DEL PUJOL, Amélie [FR/FR]; 6, square Raynouard, F–78150 Rocquencourt (FR).

(74) Mandataire: DENIS, Hervé; Bull S.A., 68, route de Versailles, F–78430 Louveciennes (FR).

(81) Etats désignés: US, brevet européen (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

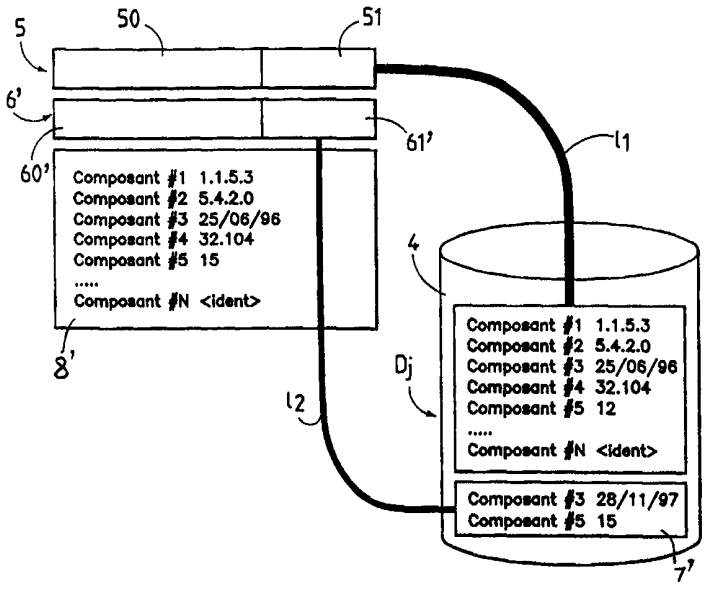
Publiée
Avec rapport de recherche internationale.
Avant l'expiration du délai prévu pour la modification des revendications, sera republiée si des modifications sont reçues.

(54) Title: METHOD FOR IDENTIFYING AND MONITORING EVOLUTION OF A SET OF SOFTWARE COMPONENTS

(54) Titre: PROCEDE D'IDENTIFICATION ET DE SUIVI DES EVOLUTIONS D'UN ENSEMBLE DE COMPOSANTS LOGICIELS

(57) Abstract

The invention concerns a method for identifying and monitoring the evolution of an set of software components (4), each component (#1 to *N*) being marked by a name and a version identifier, and each set of software components (4) being linked to a first subset (5) comprising an identifier (50, 51) of the software set and its version. The modifications brought to the software components (#1 to *N*) are marked by a correction package index which is incremented when a new correction package is applied. A second subset (6) is provided comprising an identifier (60) of the software set (4) and of the correction package, and of an evolution version number (61) comprising at least the correction package index. Said second subset (6) is associated (l_2) with a file (7') describing the name of the software components (#3, #5) of the correction package and their versions.



(57) Abrégé

L'invention concerne un procédé d'identification et de suivi des évolutions d'un ensemble de composants logiciels (4), chaque composant (#1 à #*N*) étant repérés par un nom et un identifiant de version, et chaque ensemble de composants logiciels (4) étant lié à un premier sous–ensemble (5) comprenant un identifiant (50, 51) de l'ensemble logiciel et de sa version. Les modifications apportées aux composants logiciels (#1 à #*N*) sont repérées par un indice de lot de corrections qui est incrémenté lors de l'application d'un nouveau lot de corrections. On prévoit un second sous–ensemble (6) comprenant un identifiant (60) de l'ensemble de composants logiciels (4) et du lot de corrections, et d'un numéro de version d'évolution (61) comprenant au moins l'indice de lot de corrections. Ce second sous–ensemble (6) est associé (l_2) à un fichier (7') décrivant les noms des composants logiciels (#3, #5) du lot de corrections et leurs versions.

UNIQUEMENT A TITRE D'INFORMATION

Codes utilisés pour identifier les Etats parties au PCT, sur les pages de couverture des brochures publiant des demandes internationales en vertu du PCT.

AL	Albanie	**ES**	Espagne	**LS**	Lesotho	**SI**	Slovénie
AM	Arménie	**FI**	Finlande	**LT**	Lituanie	**SK**	Slovaquie
AT	Autriche	**FR**	France	**LU**	Luxembourg	**SN**	Sénégal
AU	Australie	**GA**	Gabon	**LV**	Lettonie	**SZ**	Swaziland
AZ	Azerbaïdjan	**GB**	Royaume-Uni	**MC**	Monaco	**TD**	Tchad
BA	Bosnie-Herzégovine	**GE**	Géorgie	**MD**	République de Moldova	**TG**	Togo
BB	Barbade	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tadjikistan
BE	Belgique	**GN**	Guinée	**MK**	Ex-République yougoslave de Macédoine	**TM**	Turkménistan
BF	Burkina Faso	**GR**	Grèce			**TR**	Turquie
BG	Bulgarie	**HU**	Hongrie	**ML**	Mali	**TT**	Trinité-et-Tobago
BJ	Bénin	**IE**	Irlande	**MN**	Mongolie	**UA**	Ukraine
BR	Brésil	**IL**	Israël	**MR**	Mauritanie	**UG**	Ouganda
BY	Bélarus	**IS**	Islande	**MW**	Malawi	**US**	Etats-Unis d'Amérique
CA	Canada	**IT**	Italie	**MX**	Mexique	**UZ**	Ouzbékistan
CF	République centrafricaine	**JP**	Japon	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Pays-Bas	**YU**	Yougoslavie
CH	Suisse	**KG**	Kirghizistan	**NO**	Norvège	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	République populaire démocratique de Corée	**NZ**	Nouvelle-Zélande		
CM	Cameroun			**PL**	Pologne		
CN	Chine	**KR**	République de Corée	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Roumanie		
CZ	République tchèque	**LC**	Sainte-Lucie	**RU**	Fédération de Russie		
DE	Allemagne	**LI**	Liechtenstein	**SD**	Soudan		
DK	Danemark	**LK**	Sri Lanka	**SE**	Suède		
EE	Estonie	**LR**	Libéria	**SG**	Singapour		

PROCEDE D'IDENTIFICATION ET DE SUIVI DES EVOLUTIONS D'UN ENSEMBLE DE COMPOSANTS LOGICIELS

La présente invention concerne un procédé d'identification et de suivi des évolutions d'un ensemble de composants logiciels.

Un produit logiciel est habituellement formé d'un ensemble de composants unitaires parfaitement identifiés par une série d'informations, telles que la date de création, estampille du type de version, etc., ce qui permet de suivre leurs évolutions.

Cependant il n'existe généralement pas d'identification non ambiguë portant sur le produit logiciel considéré dans sa globalité et permettant de suivre l'évolution de chacun de ses composants.

En outre, lorsqu'on s'intéresse à une solution logicielle qui est composée d'un ensemble de produits logiciels, qui plus est peuvent être hétérogènes et multi-fournisseurs, il n'existe aucun moyen d'identifier globalement cet ensemble, ni surtout de suivre son évolution.

Or, la maîtrise de l'état de chaque composant unitaire d'une solution logicielle, qui consiste à s'assurer qu'il se trouve bien à un niveau donné (c'est-à-dire à une version donnée), est le seul moyen de garantir son intégrité.

Ce problème se pose déjà avec une certaine acuité pour les petits systèmes (micro-ordinateurs) et les systèmes moyens ("mini-ordinateurs ou ordinateurs dits "départementaux"), ce d'autant plus que la puissance de ces systèmes augmentent régulièrement. On conçoit aisément que le problème évoqué se pose avec encore plus d'acuité pour les

très gros systèmes ou "mainframe", selon la terminologie anglo-saxonne.

Aussi, le procédé de l'invention concerne plus particulièrement les systèmes de ce type et on va en rappeler brièvement les caractéristiques principales.

Ces systèmes, très complexes, intègrent de très nombreux sous-systèmes, tant matériels que logiciels qui coopèrent entre eux. Il n'est pas rare que des produits logiciels constituants de ces systèmes, comportent des centaines de milliers, voire des millions de lignes de code.

Dans ce contexte, un des problèmes les plus difficiles à résoudre est d'obtenir la cohérence et la compatibilité entre les différents composants, notamment lorsqu'ils doivent être mis à jour (passage à des versions plus récentes, ajout de modules ou de fonctions, etc.).

En effet, outre la complexité intrinsèque rappelée, on doit également tenir compte du fait que les différents produits composant le système ont des provenances disparates, que ce soit en interne (même si des règles de développement communes sont en vigueur) ou en externe (produits disponibles sur le marché).

Dans l'état de la technique, il existe deux types principaux d'architectures.

Une première architecture appartient au type dit "propriétaire". Sous ce vocable, on désigne des systèmes dont l'architecture est conçue par le constructeur de matériel informatique.

L'architecture se caractérise essentiellement par une intégration poussée et se présente sous la forme d'un système monolithique.

Dans ce système, tous les composants de l'ensemble sont pris en compte lors de l'installation initiale et lors

des mises à jour ultérieures (changement de version du système). Dans ce type d'environnement, la cohérence du système est obtenue par la validation globale des interfaces entre tous les composants. Pour l'utilisateur, ces dispositions garantissent une grande robustesse du système indépendamment de ses évolutions. En outre, elles permettent un suivi simplifié des versions successives du système. Enfin, elles assurent une disponibilité maximale et une grande sécurité de fonctionnement, puisque les risques de dysfonctionnement sont réduits au minimum.

Cependant, il existe en contrepartie des contraintes et des inconvénients sérieux.

Ceux-ci trouvent leur origine dans le fait que les systèmes sont monolithiques. Quel que soit le composant à changer et/ou la fonction à ajouter, il est nécessaire de faire évoluer tout le système. La mise à jour ne peut être que globale.

Enfin, la mise à jour entraîne une indisponibilité importante du système de traitement de données, couramment plusieurs jours pour de très gros systèmes.

Il s'ensuit que ce type d'architecture a relativement peu de souplesse et présente des possibilités d'évolution limitées.

Une seconde architecture appartient au type dit "ouvert". Sous ce vocable, on désigne des systèmes dont l'architecture n'est pas définie par un seul constructeur. Ces systèmes sont apparus, notamment, avec l'émergence des systèmes d'exploitation du type "UNIX" (marque déposée) ou similaires, par exemple "AIX" (marque déposée).

Il est clair que ce type d'architecture est très attractif, car il permet de faire appel à des logiciels hétérogènes, c'est-à-dire d'origines diverses, coexistant sur une même machine.

Pour ce type d'architecture, l'installation et/ou la mise à jour d'un logiciel ne concernent que le produit lui-même, et non le système dans sa globalité.

Puisque des composants de base peuvent être installés ou mis à jour à tout moment, une architecture "ouverte" offre donc une très grande souplesse et des facilités d'évolution importantes, du moins en théorie.

En effet, ce type d'architecture n'est pas non plus exempt d'inconvénients. En particulier, il n'offre aucune garantie de bon fonctionnement avec les autres composants logiciels : possibilités d'incompatibilité diverses, etc.

Or, malgré les inconvénients qui lui sont inhérents, un système à architecture ouverte reste très séduisant pour les utilisateurs.

En effet, il permet de satisfaire, tout ou partie, des besoins des utilisateurs de systèmes informatiques qui se font actuellement sentir, parmi lesquels :

- autonomie, c'est-à-dire la possibilité de faire évoluer un ou plusieurs produits du système, indépendamment du reste du système ;

- parallélisme, c'est-à-dire la possibilité de faire tourner plusieurs versions du même produit simultanément ;

- compatibilité, c'est-à-dire pouvoir compter sur une garantie de compatibilité ascendante des produits et des interfaces entre produits ;

- mise à jour pendant l'exploitation courante du système, c'est-à-dire pouvoir changer de version d'un produit sans arrêt de l'exploitation ou, pour le moins, avec un arrêt minimum ;

- retour en arrière, c'est-à-dire la possibilité de revenir à une version antérieure d'un produit en cas de problème, sans arrêt de l'exploitation ;

- maîtrise des évolutions, c'est-à-dire la possibilité de rester maître du processus d'évolution par la mise en oeuvre de procédures automatisées et simplifiées ;

- et rapidité des évolutions et de la maintenance, c'est-à-dire pouvoir compter sur un délai minimum entre une demande d'évolution ou de correction, et son installation effective sur site.

Pour pallier les inconvénients des deux types d'architectures précités, tout en conservant les avantages de chacun, la Demanderesse a proposé, dans une demande de brevet français N° 97 08332, déposée le 2 juillet 1997 et intitulée "Architecture de système de traitement de l'information", une solution visant à répondre pleinement aux besoins des utilisateurs qui viennent d'être rappelés.

Pour ce faire, le système de traitement de l'information présente une architecture en domaines.

Un domaine est défini comme regroupant un ensemble de fonctionnalités offertes par des composants logiciels du système. Il a des caractéristiques propres et évolue de façon indépendante. Le système est alors composé d'un ensemble de domaines ayant un minimum d'interdépendances, ce qui garantit la cohérence globale.

En d'autres termes, la souplesse d'installation et la mise à jour sont introduites à un niveau macroscopique qui reste maîtrisable, c'est-à-dire le domaine au sein duquel la cohérence reste assurée.

Chaque domaine, quel qu'il soit, est "vu" de façon identique par l'utilisateur, grâce notamment à des attributs qui lui sont affectés et un descripteur.

La figure 1 annexée à la présente description illustre schématiquement une telle architecture dite en "domaines".

Le système 1 comprend une plate-forme matérielle 10. Selon une caractéristique importante de la demande de brevet précitée, l'ensemble des logiciels et progiciels est scindé en domaines, 11 à 13. De façon plus particulière, il existe un ensemble de domaines, D_1, D_2, ..., D_j, ...,D_n, regroupés sous la référence générale 12, et deux domaines spécifiques, 11 et 13. Ces deux derniers domaines sont, respectivement, le domaine 11 fournissant le système d'exploitation (ou "OS") et les logiciels de base associés du système, et le domaine 13 fournissant le moniteur d'exploitation du système 1. Le système 1 comprend donc au minimum ces deux domaines, 11 et 13, en sus de la plate-forme matérielle 10. C'est la version du domaine 13, moniteur d'exploitation du système, qui identifie complètement l'ensemble du système 1 et de ses composants de base, c'est-à-dire les différents domaines et les fonctionnalités qu'ils offrent.

Les indices j et n repérant les domaines sont purement arbitraires, j étant un indice intermédiaire et n le nombre maximal de domaines (non compris les domaines spécifiques 11 et 13), du moins dans un état de configuration donné du système.

Les autres domaines, 12, sont optionnels et sont sélectionnés par l'utilisateur en fonction de ses besoins précis et de l'application considérée. On peut appeler ces domaines "domaines banalisés", en ce sens qu'ils sont "vus" de façon semblable par le système, même si leur contenu est différent. Le système 1 est donc entièrement modulaire au niveau des domaines.

Pour fixer les idées, les domaines suivants peuvent être implémentés dans le système : fonctions d'administration du système, fonctions de traitement par lots, fonctions de gestion de bases de données, fonction de sécurité, fonctions de gestion des processus transactionnels et fonctions d'interopérabilité.

Un système déterminé 1, répondant aux besoins de l'utilisateur, est installé dans un état donné. En d'autres termes, la configuration et les versions des domaines sont précisément définies.

Cette installation peut être obtenue de deux façons :

- Il s'agit tout d'abord de l'installation initiale proprement dite, que l'on peut appeler "clés en mains", c'est-à-dire d'une pré-configuration réalisée en usine.

- Il peut s'agir aussi d'une mise à jour incrémentale de la machine sur site d'exploitation. Les mises à jour concernent alors un ou plusieurs domaines, selon les besoins. Mais il n'est jamais nécessaire, contrairement aux systèmes de type "propriétaire" de l'art connu, de modifier tout le système 1.

Il existe en réalité deux types principaux de mise à jour :

- La mise à jour proprement dite (ou "update" selon la terminologie anglo-saxonne) : un domaine donné, D_j, ayant fait l'objet d'une installation initiale, il s'agit d'en installer une version plus récente pour y incorporer, soit des nouvelles fonctionnalités, soit des corrections. Cette opération entraîne notamment le changement de la version du domaine.

- Ajout (ou "add-on" selon la terminologie anglo-saxonne) : il s'agit de l'ajout pur et simple d'un domaine particulier, par exemple D_n, qui n'avait pas été installé initialement.

Il doit être clair également que un ou plusieurs domaines peu(ven)t être supprimé(s).

Tous les domaines, 11 à 13, coopèrent avec la plate-forme matérielle 10, directement ou via le domaine de base ou système 11. De même, tous les domaines "banalisés" 12

coopèrent avec les domaines spécifiques, 11 et 13. Des liens verticaux (sur la figure 1) illustrent ce mode de fonctionnement.

Il est avantageux que les domaines 12 présentent la plus grande indépendance les uns vis-à-vis des autres. En d'autres termes, il est important que les interactions entre ces domaines soient minimisées. Généralement, il existe cependant des relations entre certains des domaines D_1 à D_n. Ces relations ont été figurées par des liens horizontaux (sur la figure 1).

Bien que le nombre de domaines ne soit pas, *a priori*, limité, dans la pratique, le nombre raisonnable de domaines est préférentiellement inférieur à dix et, en tout cas, devrait se situer dans la gamme de dix à vingt au maximum.

Dans le cas contraire, la complexité augmente de façon très importante : fonction en $x(x-1)/2$, avec x nombre total de domaines. Les avantages apportés par l'architecture selon l'invention s'en trouvent alors largement amoindris.

En résumé, l'architecture en domaines qui vient d'être rappelée permet de cumuler les avantages des architectures de type "propriétaire" et de type "ouvert" : robustesse, disponibilité, sécurité, souplesse et évolutivité.

Elle présente donc de nombreux avantages, mais il subsiste cependant un inconvénient qui va être explicité ci-après.

Comme il a été précédemment indiqué, les architectures selon l'art connu ne permettent, pour la plupart, que la seule identification de chaque composant unitaire, ce qui rend très difficile de suivre son évolution et de connaître son état par rapport à une référence dûment qualifiée.

Par contre, l'architecture en domaines permet de bien caractériser l'état initial d'un domaine, c'est-à-dire d'un ensemble logiciel comprenant plusieurs composants, mais ne permet pas l'identification et le suivi des évolutions de cet ensemble logiciel.

En cela, malgré ses avantages certains, l'architecture en domaines laisse donc subsister un inconvénient commun aux autres architectures.

L'invention vise à pallier les inconvénients des procédés de l'art connu.

Elle se fixe pour buts :

- d'identifier l'état d'un ensemble de produits logiciels ;
- de vérifier la cohérence de cet ensemble par rapport à une référence ;
- et de suivre l'évolution de cet ensemble.

Pour ce faire, le procédé selon l'invention consiste à prévoir une structure autorisant la gestion et le suivi de l'évolution d'un ensemble logiciel cohérent.

Cette structure comprend un composant unitaire consistant en un enregistrement lié à l'ensemble logiciel précité et parfaitement identifiable par le système, dont la configuration sera détaillée ci-après. On rattache notamment à ce composant un ensemble de fichiers autorisant le suivi des évolutions et un fichier de référence contenant les identifications unitaires de chaque composant modifié.

Bien que le procédé selon l'invention s'applique de façon préférentielle à une architecture en domaines telle qu'elle vient d'être rappelée, il ne saurait être limité à ce seul type d'architecture.

Cependant, pour fixer les idées, le procédé de l'invention sera décrit ci-après dans le cadre de cette application préférée, sauf mention contraire.

L'invention a donc pour objet un procédé d'identification et de suivi des évolutions d'un ensemble de composants logiciels déterminé dans un système de traitement de l'information, chaque composant étant repéré par un nom et un identifiant de version, et chaque ensemble de composants logiciels étant lié à un premier sous-ensemble comprenant un identifiant dudit ensemble logiciel et de sa version, caractérisé en ce que les modifications apportées à tout ou partie desdits composants d'un ensemble de composants logiciels déterminé sont repérées par un indice dit de lot de corrections, en ce que l'ensemble de composants logiciels déterminé est associé à un second sous-ensemble comprenant un identifiant dudit ensemble de composants logiciel déterminé et dudit lot de corrections et d'un identifiant de version d'évolution, cet identifiant de version d'évolution comprenant au moins ledit indice de lot de correction, et en ce que ledit second sous-ensemble est lié à au moins une première suite d'enregistrements décrivant les noms des composants dudit lot de corrections et leurs versions.

L'invention sera mieux comprise et d'autres caractéristiques et avantages apparaîtront à la lecture de la description qui suit en référence aux figures annexées, parmi lesquelles :

- la figure 1 illustre schématiquement une architecture de système de traitement de l'information en domaines ;

- la figure 2 illustre schématiquement la structure d'un domaine, élément constitutif de base de l'architecture de la figure 1 ;

- la figure 3 illustre un exemple d'organisation hiérarchique des domaines ;

- la figure 4 illustre un domaine particulier et ses relations avec les autres domaines du système ;

- les figures 5a à 5c illustrent schématiquement le procédé conforme à l'invention.

Le procédé selon l'invention va être décrit, comme il a été indiqué dans le cadre de l'application préférée de l'invention, c'est-à-dire dans le cadre d'un système de traitement de l'information à architecture en domaines.

Aussi, il apparaît tout d'abord utile de décrire, de façon plus détaillée, par référence à la figure 2, une unité élémentaire que constitue un domaine, Dj, d'indice arbitraire j.

On peut définir un domaine Dj comme étant une unité indépendante d'intégration, d'installation et de mise à jour. Elle comprend un ou plusieurs logiciels ou progiciels de provenances diverses : interne ou externe, que l'on appellera produits. Par "interne", on entend des applications de type "propriétaire". Par "externe", on entend des applications disponibles sur le marché.

De façon plus précise, un domaine comprend deux parties principales.

Une première partie 3 est constituée des différents produits offrant des fonctionnalités propres à un domaine, repérées 30 à 32. A titre d'exemple, un domaine de traitement par lot offrira au moins les fonctions suivantes : gestions des impressions et des sauvegardes.

La seconde partie 2 constitue une interface particulière du domaine Dj, que l'on peut appeler "descripteur" qui donne accès à des données décrivant le domaine et ses propriétés.

Les fonctionnalités fournies par les domaines Dj sont supportées par des produits qui se caractérisent, entre

autres, par la manière dont ils sont intégrés dans le système 1, par la façon dont ils s'installent et s'exécutent et par les autres produits (c'est-à-dire les autres domaines) dont ils dépendent éventuellement.

La partie 2 ou descripteur doit donc donner accès à des informations caractérisant un domaine particulier D_j, notamment à un identifiant concernant son nom et sa version, et à une liste de ses composants. En outre, un domaine D_j est caractérisé par un certain nombre d'attributs, comme il a été indiqué. Toutes ces informations sont nécessaires pour pouvoir réaliser les opérations de base susmentionnées : installation et mise à jour des domaines. Elles décrivent aussi le mode de fonctionnement des produits composant le domaine.

Dans un mode de réalisation préféré, les attributs sont au nombre de trois, à savoir le niveau d'insertion, le type d'occurrence et les dépendances.

On va maintenant décrire de façon détaillée un exemple d'organisation pratique d'un domaine D_j, par référence à la figure 3.

Dans cet exemple pratique, le domaine est organisé selon une hiérarchie d'objets.

On va considérer ci-après, pour des raisons de simplification, un domaine D_j ne contenant qu'un seul produit P_i, étant entendu que ces dispositions s'appliquent tout aussi bien à des domaines comprenant plusieurs produits. Il est en effet clair que la structure de base d'un domaine est le produit, un domaine comprenant un ou plusieurs produits.

On peut cependant distinguer une structure hiérarchique à quatre niveaux, dans un mode de réalisation avantageux.

Le premier niveau est le produit P_i proprement dit. C'est un ensemble homogène offrant une fonctionnalité donnée. Le produit est constitué d'un ensemble de modules M_{i1} à M_{i3} (trois dans l'exemple décrit sur la figure 3) ou "packages", qui constituent le deuxième niveau hiérarchique.

Le deuxième niveau regroupe donc les modules M_{i1} à M_{i3}. Un module M_{i1} à M_{i3}, est un sous-ensemble fonctionnel homogène d'un produit P_i. Il est constitué d'un ensemble de jeux de fichiers ou "filesets" FS_{i1}, FS_{i2} et FS_{i31} et FS_{i32} (dans l'exemple décrit sur la figure 3).

Ces deux niveaux constituent un premier niveau de visibilité externe, NVe, c'est-à-dire "visible" par l'utilisateur pour une mise à jour du domaine D_j et/ou son installation initiale.

Il est ainsi possible d'installer tout ou partie des produits d'un domaine Dj, ainsi que tout ou partie des modules les composant, en tenant compte naturellement de contraintes de cohérence liées à ces produits.

Le troisième niveau regroupe les jeux de fichiers FS_{i1}, FS_{i2} et FS_{i31} et FS_{i32}. Un jeu de fichiers est un ensemble de fichiers contribuant à la réalisation de tout ou partie des fonctionnalités offertes par un module. Un module, par exemple les modules M_{i1} et M_{i2}, peut ne comprendre qu'un seul jeu de fichiers, FS_{i1} et FS_{i2}, respectivement. Au contraire, un module peut comprendre deux jeux de fichiers ou plus. C'est le cas du module M_{i3} qui comprend deux jeux de fichiers, FS_{i31} et FS_{i32}.

Le quatrième niveau est constitué par les fichiers eux-mêmes, F_{i11}, F_{i12}, F_{i21}, F_{i22}, F_{i31}, F_{i321} et F_{i322}. Chaque jeu de fichiers peut comprendre un ou plusieurs fichiers.

Les troisième et quatrième niveaux constituent un niveau de visibilité interne NVi, non visible par

l'utilisateur en utilisation normale. Cependant, les mises à jour concernent ce niveau.

Un domaine D_j est donc un ensemble de produits et de modules. Il comprend toujours, au minimum, un module et un jeu de fichiers spécifiques décrivant le domaine D_j. Ce jeu de fichiers spécifiques comprend, au moins, un fichier également spécifique contenant la liste, c'est-à-dire le nom et le numéro de version, de tous les autres jeux de fichiers du domaine D_j. Outre son rôle de conteneur, ce fichier particulier sert à assurer la cohérence du domaine D_j. En effet, tous les autres jeux de fichiers du domaine D_j sont déclarés dépendants du jeu de fichiers spécifiques. L'installation ou la mise à jour d'un jeu de fichiers quelconque, appartenant au domaine D_j, n'est possible que si la version du domaine le permet, du moins en ce qui concerne les produits contrôlés ou intégrés, c'est-à-dire d'un niveau d'insertion suffisamment élevé.

Enfin, dans un mode de réalisation particulier d'une architecture en domaine, illustré schématiquement par la figure 4, le système 1 peut comprendre un troisième domaine spécifique, 14, que l'on peut appeler domaine de service ou domaine "outils".

Ce domaine comprend, notamment, des produits logiciels ou des utilitaires, également spécifiques, permettant l'installation et/ou la mise à jour de tout ou partie des domaines, D_1 à D_n, et dans chaque domaine, de tout ou partie des produits, qu'ils soient d'origine interne, c'est-à-dire *a priori* entièrement intégrés aux domaines, ou externe, ces produits externes étant contrôlés ou non, selon le niveau d'insertion associé au domaine. Naturellement, les outils, ou "moteurs", d'installation et de mise à jour présents dans ce domaine sont en interaction avec les domaines, D_1 à D_n, ainsi qu'avec le domaine de base 11, et le domaine moniteur d'exploitation 13, et plus particulièrement avec leurs ensembles descripteurs 2. Les

installations s'effectuent selon des règles enregistrées dans le domaine 14 et/ou les ensembles descripteurs 2.

Si on se reporte de nouveau à la figure 2, ce qui a été appelé descripteur 2 donne accès à, au moins, un enregistrement 20, contenant le nom et la version du domaine Dj, à un enregistrement 21, décrivant les attributs associés à ce domaine et à un enregistrement 22, de description du contenu du domaine et à un jeu de règles d'installation et de mise à jour du domaine et des produits le composant. Comme il a été décrit en regard de la figure 3, chaque domaine Dj comporte un jeu de fichiers spécifique contenant la liste des autres jeux de fichiers, c'est-à-dire au moins leurs noms et leurs numéros de version. On peut également compléter ces informations par des scripts et des règles d'installation et de mise à jour des produits externes au domaine.

Bien que les informations rappelées ci-dessus puissent être réparties à l'intérieur d'un domaine donné, celui-ci est "vu" de l'extérieur de façon identique, via l'ensemble descripteur 2 qui constitue une interface standard pour tous les domaines, même si le contenu de l'information accédée diffère d'un domaine à l'autre.

On va maintenant décrire le procédé selon l'invention par référence aux figures 5a à 5c.

En résumé de ce qui vient d'être rappelé en relation avec une architecture en domaines, on constate que chaque domaine Dj possède une structure qui permet de l'identifier et d'identifier ses composants logiciels : nom du domaine, numéro de version, fichier de référence descriptif des composants unitaires du domaine. C'est le rôle joué par le descriptif 2 (figure 2), plus particulièrement des enregistrements 20 (nom et version du domaine) et 22 (description du contenu du domaine), ainsi que du jeu de fichiers spécifique contenant la liste des autres fichiers (noms et numéros de version).

A titre d'exemple pratique, le numéro de version est associé à différents indices structurés qui peuvent se présenter sous la forme suivante : "v.r.m". "v" est un indice concernant la version livrée du domaine Dj. "r" est un indice dit de "release", c'est-à-dire de mise à disposition. Cet indice "r" indique des versions améliorées, mais sans changement fonctionnel majeur. L'indice "m" est relatif à la re-livraison de tout élément du domaine, fonctionnel ou non. Il est incrémenté à cette occasion, en général à partir de la valeur zéro (état initial).

Toutes ces dispositions permettent de bien caractériser l'état initial d'un domaine particulier Dj, avec tous ses composants, ce qui constitue un avantage par rapport aux autre architectures.

Par contre, elles ne permettent pas de réaliser correctement le suivi des évolutions de ce domaine particulier Dj.

Aussi, selon une première caractéristique importante de l'invention, on ajoute à la structure existante un composant unitaire supplémentaire qui va permettre ce suivi, qui constitue un sous-ensemble du domaine.

Selon une deuxième caractéristique importante de l'invention on utilise un indice supplémentaire représentatif des évolutions du domaine Dj, que l'on appellera ci-après indice "f" et qui est relatif aux lots de correction des domaines. Cet indice peut être avantageusement numérique, ce qui permet de le modifier simplement par incrémentation.

Sur la figure 5a, on a représenté un des domaines présents dans le système 1 de la figure 1 : le domaine Dj, d'indice arbitraire *j*. On suppose qu'il s'agit de l'état initial du domaine Dj, c'est-à-dire l'état dans lequel il a été livré pour sa première installation sur le système 1 (figure 1).

On suppose que le domaine D*j* comprend *N* composants de diverses natures, référencés sur la figure 5a "composant #1" à "composant #*N*". Ces composants sont associés à des indices de version qui peuvent être de structures diverses : date, numéros de version conformes à la structure précitée, à laquelle on a ajouté un quatrième indice (ce qui conduit à une structure similaire à "v.r.m.f") ou tout autre mode d'identification de version. À titre d'exemple, la version des composants "#1" et "#2" est repérée par une suite structurée du type "v.r.m.f" ("1.1.5.3" et "5.4.2.0", respectivement), le composant "#3" par une date et les composants #4" et "#5" par de simples numéros ("32.104" et "12", respectivement). Le composant numéro "#*N*" par un mode non explicitement précisé (référencé "<ident>"). A priori, il n'y a pas de règles formelles pour identifier la version des composants. Toutes méthodes d'identification n'introduisant pas d'ambiguïté peut être utilisée.

L'ensemble des composants "composant #1" à "composant #*N*", repéré 4 sur la figure 5a, constitue un ensemble logiciel dont on veut suivre l'évolution dans le temps, pour en garantir la cohérence par rapport à une référence donnée.

Dans le cas d'une architecture en domaines, qui constitue une architecture préférentielle, l'ensemble logiciel 4 peut être assimilé à la partie 3 (figure 2) d'un domaine D*j*.

On associe (lien figuré par un trait plein l_1) à l'ensemble logiciel proprement dit 4, un premier composant ou sous-ensemble 5 identifiant cet ensemble logiciel, par exemple le domaine D*j*. Ce composant est un enregistrement comprenant deux parties : une partie "nom" 50, et une partie "numéro ou identifiant de version de domaine" 51 (et de façon plus générale, d'ensemble logiciel).

A titre d'exemple, le nom 50 peut avoir la structure composée suivante "<nom_domaine>.saga_Id", dans laquelle

"<nom_domaine>" est un nom de domaine arbitraire et "saga_Id" un suffixe permettant d'identifier un objet de type domaine, et de façon plus générale l'appartenance à un type d'entité logicielle déterminée.

Le numéro de version 51 aura avantageusement une structure similaire à la structure "v.r.m.f" précitée. *A priori*, le quatrième indice élémentaire est égal à zéro. Dans l'exemple illustré, le numéro de version du domaine 51 est "3.2.1.0".

Un deuxième composant ou sous-ensemble 6 est associé à l'ensemble logiciel 4, il s'agit d'un composant entièrement spécifique à l'invention. Ce composant 6 est un enregistrement identifiant l'évolution de l'ensemble logiciel 4. La structure de ce composant est similaire à celle du composant 5. Elle comprend une partie "nom" 60 et une partie "numéro de version d'évolution" 61, cette dernière étant conforme à la structure "v.r.m.f" précitée. L'indice partiel principal, caractéristique de cette structure, est l'indice supplémentaire "f" indiquant le numéro du lot de correction. La partie nom 60 permet de lier l'enregistrement à un ensemble logiciel donné 4, notamment à un domaine dans l'application préférée. Elle a la structure suivante : "<nom_domaine>.<label>", dans laquelle "<nom_domaine>" est le nom du domaine, et plus généralement le nom de l'ensemble logiciel, et "<label>" un suffixe qui permet d'identifier le composant supplémentaire, c'est-à-dire le type d'enregistrement 6.

A priori, à l'état initial, c'est-à-dire l'état représenté sur la figure 5a, l'indice élémentaire "f" (numéro du lot de correction) est égal à zéro et le "numéro de version d'évolution" 61 est identique au numéro de version de domaine 51, soit "3.2.1.0" dans l'exemple illustré.

On a représenté sous la référence 8, l'état courant de la machine, et plus particulièrement de la composition de

chaque ensemble logiciel 4. Cet état courant comprend au moins des entrées identifiant les composants élémentaires : "composant #1" à "composant #*N*", ainsi que leurs numéros de version. Dans le cas d'une architecture en domaines, qui constitue l'architecture préférentielle, si on se reporte de nouveau à la figure 4, l'état courant de la machine est géré, a priori, par le moniteur d'exploitation 13. Dans une architecture autre que l'architecture en domaine, cet état de la machine est géré par une entité similaire ou par le système d'exploitation.

On va maintenant illustrer le processus d'identification et de suivi des corrections apportées aux composants du domaine D*j*, qui porte le nom arbitraire de "nom_domaine", par référence plus particulière aux exemples illustrés par les figures 5b et 5c. Le numéro du lot de corrections (version courante) est repéré par l'indice f = 1 (cas de la première version de lot de correction : figure 5b).

Dans un premier temps, on suppose que les composants numéros "#3" et "#5" sont modifiés, c'est-à-dire mis à jour.

Le composant "#3", dont l'identifiant de version est une date (anciennement "25/06/96" dans l'exemple décrit), a été mis à jour. La date de la nouvelle version est la suivante : "28/11/97".

Le composant "#5" dont l'identifiant de version est un simple numéro de version (anciennement "12") a été également mis à jour et son nouveau numéro de version est "15".

L'enregistrement 5 n'évolue pas, car le nombre et la répartition des composants de l'ensemble logiciel 4 n'ont pas été modifiés, seules les versions de tout ou partie des composants ont été changées (en l'occurrence les composants "#3" et "#5").

Par contre l'identifiant d'évolution, c'est-à-dire l'enregistrement identifiant les corrections apportées à l'ensemble logiciel 4, référencé désormais 6' (avec ses deux parties 60' et 61'), va être mis à jour et notamment l'indice "f" précité (numéro de lot de corrections) va être incrémenté d'une unité. Le numéro de version d'évolution 61' devient donc "3.2.1.1", dans l'exemple décrit.

Selon une caractéristique supplémentaire de l'invention, un lot de corrections donné (repéré par l'indice "f" précité) est accompagné d'un fichier 7', ou suite d'enregistrements, qui identifie les noms des composants du lot de corrections apporté et leur version.

En outre, de façon avantageuse, à chaque modification de l'enregistrement 6, correspond un fichier 7' différent, ce qui permet de conserver l'historique des corrections apportées aux composants de l'ensemble logiciel 4, c'est-à-dire son évolution dans le temps depuis son état initial (figure 5a). Ces fichiers seront référencés 7' (figure 5b), 7" (figure 5c), etc.

Un lien l_2 (trait plein) symbolise la mise à jour du composant 6 (et notamment de la partie 61') et la création d'un fichier 7' associé au composant 6'.

Dans le cas du lot de corrections d'indice f = 1, illustré par la figure 5b, le fichier 7' comprend des entrées pour les composants #3 et #5, qui seuls ont été modifiés par rapport à l'état initial de l'ensemble logiciel 4. Les composants mis à jour, #3 et #5, voient leur identifiant de version mis en concordance avec le nouvel état atteint : une nouvelle date de version pour le composant "#3" et un nouveau numéro de version pour le composant "#5".

Au nouvel enregistrement 6', va correspondre aussi une mise à jour de l'état 8' de l'ensemble logiciel 4.

La figure 5c illustre une nouvelle modification de tout ou partie des composants de l'ensemble logiciel 4 (lot de corrections f = 2), en l'occurrence les composants "#1" et, de nouveau, le composant "#5". Le composant "#1" passe du numéro de version "1.1.5.3" au numéro de version "1.2.0.0" et le composant "#5" du numéro de version "15" au numéro de version "16".

Comme précédemment, l'enregistrement 5 reste inchangé et conserve le numéro de version d'ensemble logiciel "3.1.2.0" dans l'exemple décrit.

L'enregistrement identifiant l'évolution devient 6" (avec ses deux parties 60" et 61"), l'indice "f" étant de nouveau incrémenté d'une unité. Le numéro d'identifiant d'évolution 61" devient "3.2.1.2" dans l'exemple décrit.

De même, on créé un nouveau fichier 7" qui identifie les noms des composants du lot de corrections apporté (indice f = 2) et leurs versions. Dans le cas illustré par la figure 5c, les composants mis à jour étant "#1" et "#5", le fichier 7" comprend des entrées pour ces deux composants, qui seuls ont évolués par rapport à la modification précédente (figure 5b : indice f = 1).

Comme précédemment, au nouvel enregistrement 6", va correspondre aussi une mise à jour de l'état 8" de l'ensemble logiciel 4.

En résumé, lors de l'installation d'un ensemble logiciel 4 (plus particulièrement, dans l'application préférée, lors de l'installation d'un domaine Dj), le composant supplémentaire 6 conforme à l'invention est implémenté en même temps que cet ensemble logiciel.

Ensuite, chaque ensemble cohérent de modifications (lots de corrections f = x, avec x = 1, 2, etc.) est décrit dans un ensemble de fichiers 7', 7", etc., chacun étant attaché à une nouvelle version du composant spécifique à

l'invention, c'est-à-dire les enregistrements 6', 6", etc. Les fichiers 7', 7", etc., comprennent des enregistrements avec les noms des composants du lot de corrections et leurs versions.

La première version des modifications est identifiée par un numéro d'évolution de version "v.r.m.1" (la version initiale étant "v.r.m.0") et les suivantes par incrémentation de "f" (f=2, 3, 4, etc.).

L'application sur site d'un ensemble cohérent de modifications (lot de corrections) sur un système 1 (figure 1) est toujours accompagnée d'une nouvelle version de l'enregistrement 6, et plus particulièrement de la partie 61 (l'indice "f" étant incrémenté d'une unité), ce qui constitue une signature électronique de la modification apportée.

Tous les ensembles de modifications sont disjoints les uns des autres : la version "f+1" d'un ensemble de corrections ne contient pas les corrections de la version précédente "f".

Chaque ensemble de corrections possède son propre ensemble de fichiers descriptifs, 7', 7", etc. Ainsi l'historique complet des différentes évolutions restent tracées sur le site client. En d'autres termes tous les fichiers intermédiaires restent disponibles et permettent de reconstituer l'évolution complète de l'ensemble logiciel.

Enfin, jusqu'à présent, on a considéré, au moins implicitement, que les fichiers successifs 7', 7", etc. ne contenaient que des informations sur les noms de composants et sur les versions successives de ceux-ci.

Dans une variante préférée de l'invention, ces fichiers sont complétés avantageusement par des fichiers en mode texte (non représentés) documentant chacune des modifications apportées aux composants.

Les opérations de comparaison de l'état courant, 8'ou 8", de l'ensemble logiciel 4 présent sur le système 1 (selon le niveau d'évolution atteint par l'ensemble logiciel 4 et déterminé par l'indice "f") par rapport au fichier de référence joint à chaque nouvel ensemble de corrections (fichier correspondant à l'indice "f+1") peuvent être simplifiées en faisant appel à des outils d'administration spécialisés externes ou internes au système 1. En particulier, ces outils peuvent faire partie avantageusement du domaine spécifique 14, illustré par la figure 4, ou domaine de service (dans la mesure où ce domaine existe sur le système 1). Dans une variante supplémentaire, les opérations peuvent être laissées sous le contrôle du domaine de base, ou système d'exploitation ("OS") 11 (figure 1), qui existe sur toute machine.

Le procédé selon l'invention permet de suivre également les évolutions d'un nouveau domaine (ou de façon plus générale d'un nouvel ensemble logiciel). En effet, le composant supplémentaire 6 étant lié à un domaine particulier, si on ajoute un domaine, par exemple un domaine D_p (avec p arbitraire), ce composant 6 est implémenté en même temps que le nouveau domaine D_p. On considère qu'il s'agit d'un état initial et l'indice "f" prend la valeur zéro lors de la création du domaine D_p.

Lors des évolutions éventuelles ultérieures du domaine, le processus détaillé ci-dessus est réalisé et il est inutile de le redécrire.

De même la suppression d'un domaine quelconque (ou de façon plus générale d'un ensemble logiciel), par exemple le domaine D_p précité, ne pose pas de problèmes de suivi non plus, toujours pour les mêmes raisons. Le composant supplémentaire 6 étant lié à ce domaine D_p, il disparaît avec ce domaine. Toutefois, les fichiers successifs 7', 7", etc., peuvent rester enregistrés dans le système 1, à titre d'historique.

Dans les deux cas, c'est le moniteur d'exploitation, c'est-à-dire le domaine 13, pour l'architecture en domaines de la figure 1 qui identifie complètement l'ensemble du système 1 et ses composants de base, c'est-à-dire les domaines D_1 à D_n.

Enfin, lorsque la composition de l'un des domaines (ou de façon plus générale d'un ensemble logiciel), par exemple le domaine D_j, est modifiée, non pas par simple correction ou changement de version d'un composant, mais par ajout ou suppression d'un ou plusieurs composants, il est nécessaire de faire évoluer la version du domaine : numéro de version du domaine 51, de l'enregistrement 5.

A la lecture de ce qui précède, on constate aisément que l'invention atteint bien les buts qu'elle s'est fixés.

Il doit être clair cependant que l'invention n'est pas limitée aux seuls exemples de réalisations explicitement décrits, notamment en relation avec les figures 5a à 5c.

En particulier, la numérotation des versions des composants, des domaines et des évolutions peut être réalisée de diverses façons. Seul l'indice élémentaire, qui a été appelé de façon arbitraire "f" concerne directement le procédé de l'invention, car identifiant le niveau d'évolution des corrections apportées aux composants de l'ensemble logiciel. Il peut cependant être remplacé par un moyen similaire permettant de refléter des états successifs, que ce moyen soit purement logiciel (octet, etc.) ou matériel (compteur incrémenté). De même, s'il est usuel de forcer l'indice précité ou son équivalent à la valeur zéro pour identifier un état initial, cette convention est purement arbitraire et d'autres conventions peuvent être adoptées sans sortir du cadre de l'invention. On pourrait notamment utiliser les lettres de l'alphabet, au lieu d'un indice numérique ou tout autre convention établissant une relation biunivoque entre l'indice utilisé et le lot de correction.

Il doit être clair aussi que, bien que particulièrement adaptée à une architecture de système en domaines telle qu'elle a été définie dans la présente description, on ne saurait cantonner l'invention à ce seul type d'architecture. Elle s'applique au suivi de tout ensemble logiciel multicomposants.

REVENDICATIONS

1. Procédé d'identification et de suivi des évolutions d'un ensemble de composants logiciels déterminé (4) dans un système de traitement de l'information (1), chaque composant (#1 à #*N*) étant repéré par un nom et un identifiant de version, et chaque ensemble de composants logiciels (4) étant lié (l_1) à un premier sous-ensemble (5) comprenant un identifiant (50, 51) dudit ensemble logiciel et de sa version, caractérisé en ce que les modifications apportées à tout ou partie desdits composants (#1 à #*N*) d'un ensemble de composants logiciels déterminé (4) sont repérées par un indice dit de lot de corrections, en ce que l'ensemble de composants logiciels déterminé (4) est associé à un second sous-ensemble (6, 6', 6") comprenant un identifiant (60, 60', 60") dudit ensemble de composants logiciel déterminé (4) et dudit lot de corrections et d'un identifiant de version d'évolution (61, 61', 61"), cet identifiant de version d'évolution (61, 61', 61") comprenant au moins ledit indice de lot de correction, et en ce que ledit second sous-ensemble (6', 6") est lié (l_2) à au moins une première suite d'enregistrements (7', 7") décrivant les noms des composants dudit lot de corrections (#3-#5, #1-#5) et leurs versions.

2. Procédé selon la revendication 1, caractérisé en ce que, lors de l'implantation d'un nouveau ensemble de composants logiciels (4) sur ledit système de traitement de l'information (1), ledit second sous-ensemble associé (6) est implanté en même temps que celui-ci, et en ce que, ledit indice de lot de corrections étant un indice numérique, l'état initial dudit ensemble de composants logiciels déterminé (4) est repéré par la mise à la valeur zéro de cet indice.

3. Procédé selon la revendication 2, caractérisé en ce que ledit indice de lot de corrections est incrémenté d'une unité lors de la mise en place d'un nouveau lot de corrections modifiant tout ou partie desdits composants logiciels (#1 à #*N*).

4. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que, pour chaque série de modifications de tout ou partie desdits composants logiciels (#1 à #*N*) correspondant à un nouveau lot de corrections, ledit identifiant de version d'évolution (61', 61") de ce dernier est modifié.

5. Procédé selon la revendication 4, caractérisé en ce que, pour chaque série de modifications de tout ou partie desdits composants (#1 à #*N*) correspondant à un nouveau lot de corrections, il est créé une nouvelle première suite d'enregistrements (7', 7"), associée audit second sous-ensemble modifié (6', 6") et identifiant lesdits composants logiciels modifiés et leurs nouvelles versions, cette nouvelle première suite d'enregistrements (7', 7") étant disjointe des versions précédentes.

6. Procédé selon la revendication 5, caractérisé en ce que toutes lesdites suites d'enregistrements créées (7', 7") restent stockées sur ledit système de traitement d'informations (1), de manière à constituer un historique des lots de corrections successivement appliqués audit ensemble de composants logiciels déterminé (4).

7. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que ladite première suite d'enregistrements (7', 7") est associée à une seconde suite d'enregistrements comprenant des données

informationnelles sur lesdites corrections appliquées par lesdits lots de corrections.

8. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que ledit identifiant (60, 60', 60") de l'ensemble de composants logiciels déterminé (4) et du lot de corrections dudit second sous-ensemble (6, 6', 6") est structuré de la façon suivante : <nom>.<label>, structure dans laquelle <nom> identifie le nom dudit ensemble de composants logiciels (4) et <label> un suffixe indiquant l'appartenance dudit second sous-ensemble (6') à un type d'entité logicielle déterminée.

9. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que ledit numéro de version d'évolution comprend un numéro de version dudit ensemble de composants logiciels déterminé (4) complété par ledit indice de lot de corrections.

10. Procédé selon l'une quelconque des revendications précédentes, caractérisé en ce que lesdits ensembles de composants logiciels (4) sont constitués d'entités dites domaines (Dj) comprenant chacune au moins un desdits composants logiciels, en ce que chaque domaine (11-14) contient un sous-ensemble dit descripteur (2) donnant accès à des informations spécifiques déterminées (20-22), lesdites informations comprenant au moins un identifiant (20) du domaine et des données (22) décrivant lesdits composants, l'installation et/ou la mise à jour desdits domaines (11-14) étant réalisées à partir de règles prédéterminées et desdites informations spécifiques déterminées (20-22), et en ce que lesdits seconds sous-ensembles (6', 6") identifient lesdits domaines (Dj) auxquels ils sont liés et la version du lot de corrections courante appliquée à ces domaines (Dj).

11. Procédé selon la revendication 10, caractérisé en ce que ledit identifiant (20) comprend au moins des informations de nom et de version de domaine (D_j).

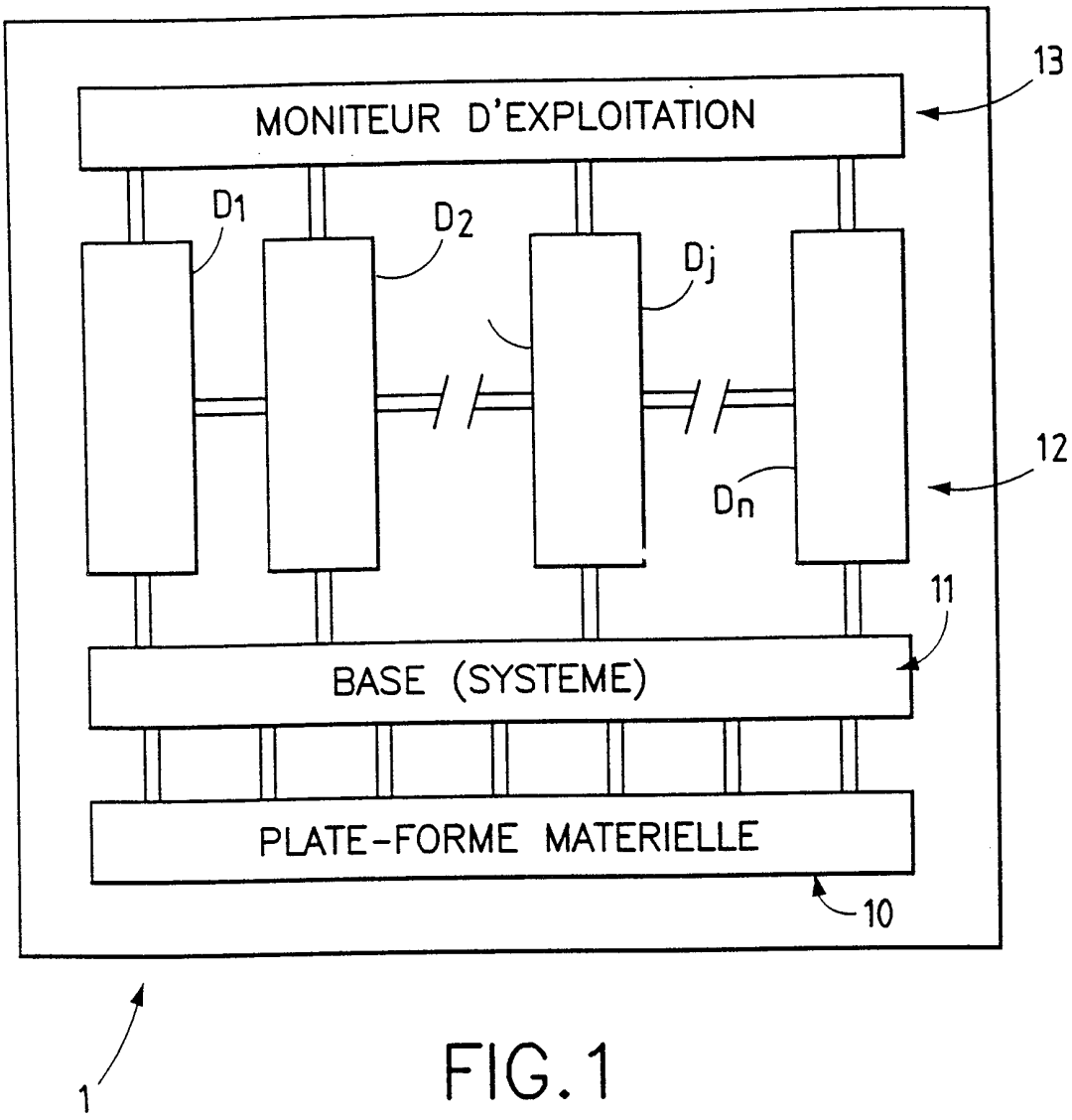

MONITEUR D'EXPLOITATION
13
D1
D2
Dj
Dn
12
11
BASE (SYSTEME)
PLATE-FORME MATERIELLE
10
1

FIG.1

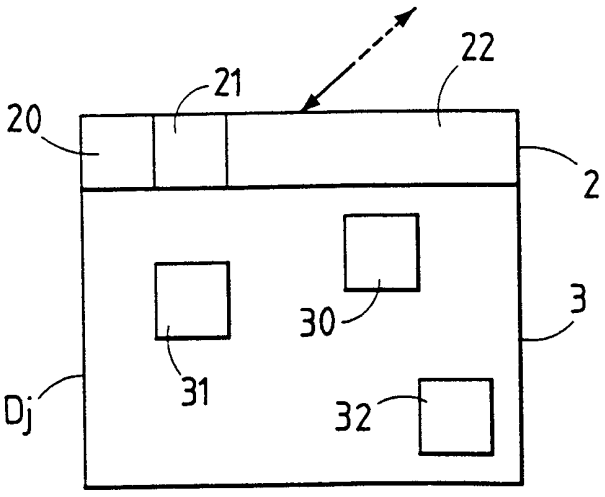

20
21
22
2
3
30
31
32
Dj

FIG.2

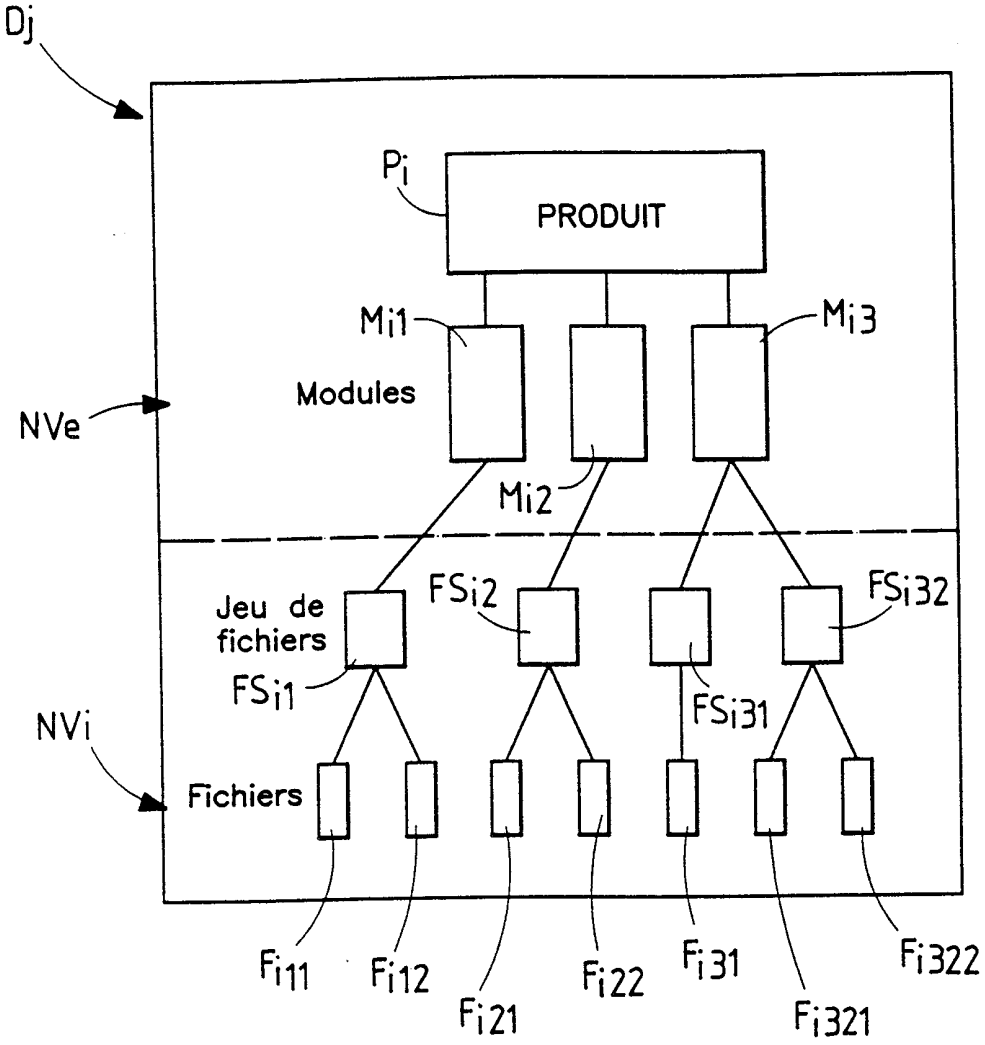

Dj
Pi
PRODUIT
Mi1
Mi3
Modules
NVe
Mi2
Jeu de fichiers
FSi2
FSi32
FSi1
FSi31
NVi
Fichiers
Fi11
Fi12
Fi21
Fi22
Fi31
Fi321
Fi322

FIG.3

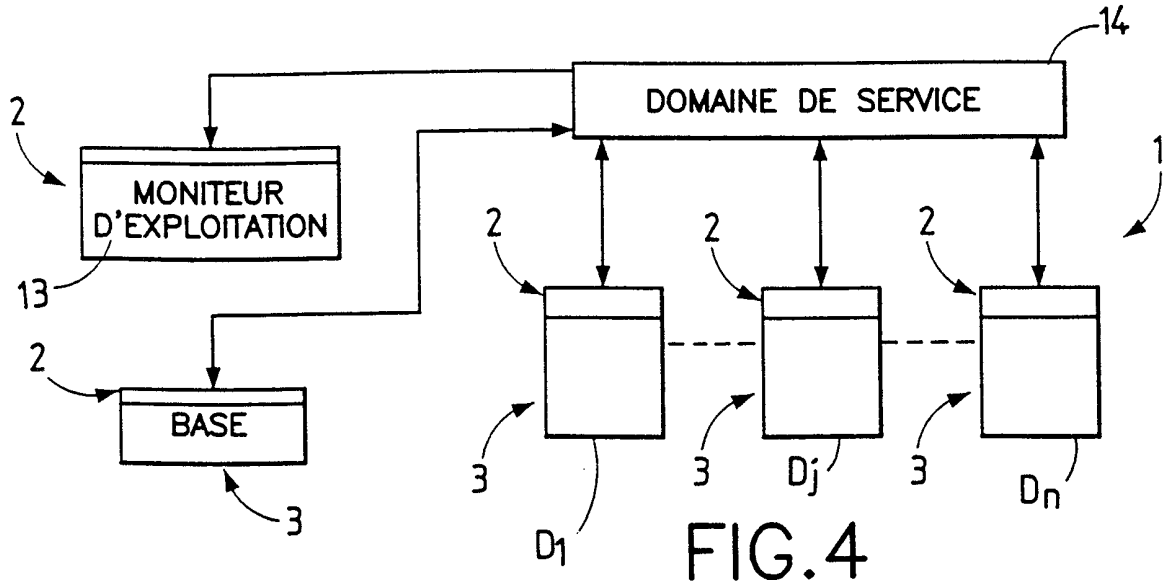

14
DOMAINE DE SERVICE
2
MONITEUR D'EXPLOITATION
13
1
2
2
2
2
BASE
3
3
3
3
Dj
Dn
D1

FIG.4

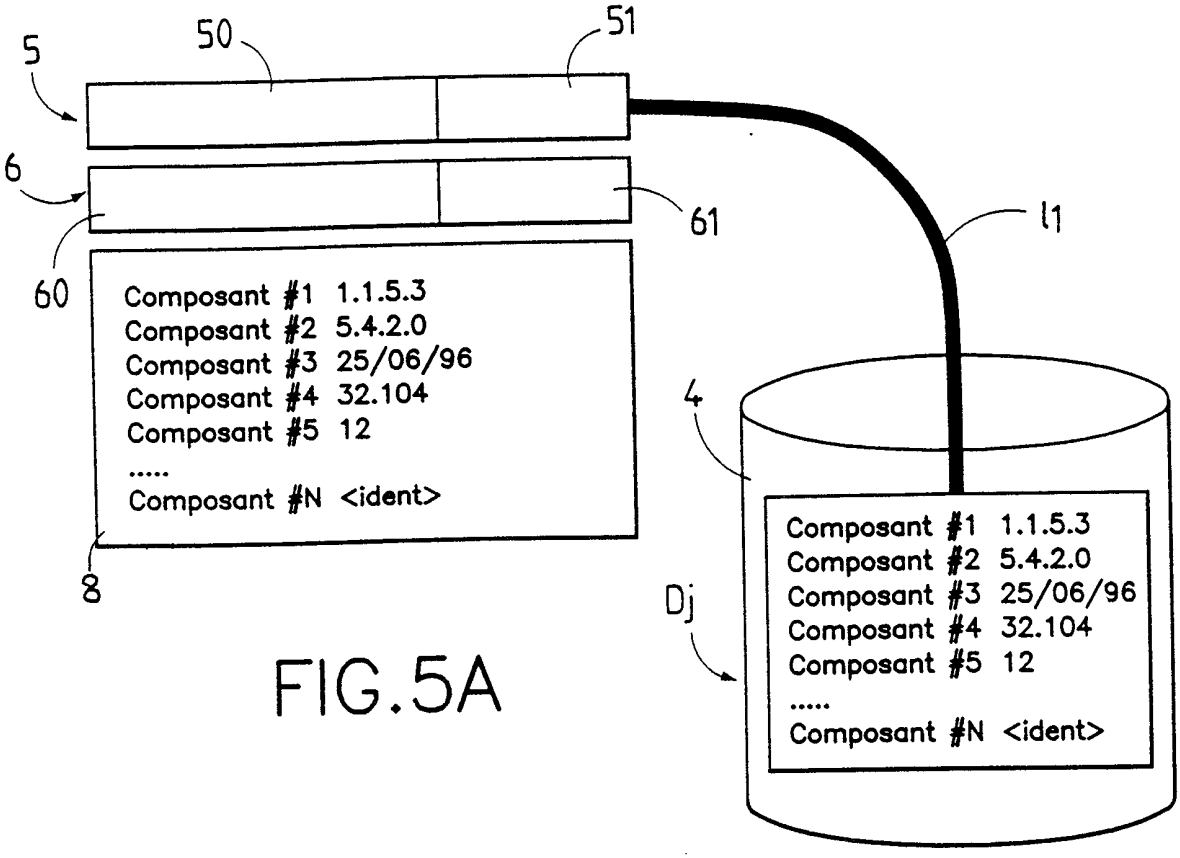

5
50
51
6
61
l1
60
Composant #1 1.1.5.3
Composant #2 5.4.2.0
Composant #3 25/06/96
Composant #4 32.104
Composant #5 12
.....
Composant #N <ident>
8
4
Dj
Composant #1 1.1.5.3
Composant #2 5.4.2.0
Composant #3 25/06/96
Composant #4 32.104
Composant #5 12
.....
Composant #N <ident>

FIG.5A

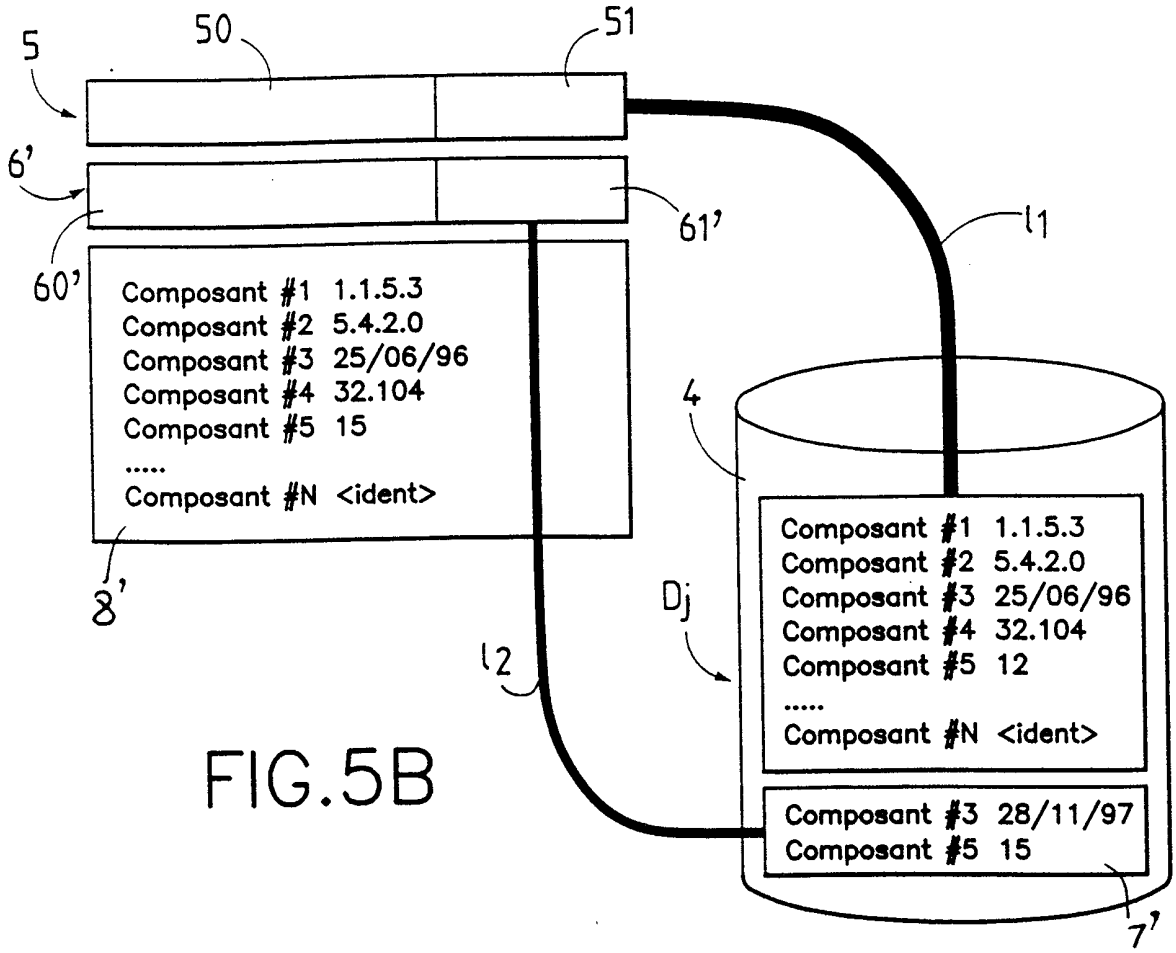

5
50
51
6'
61'
l1
60'
Composant #1 1.1.5.3
Composant #2 5.4.2.0
Composant #3 25/06/96
Composant #4 32.104
Composant #5 15
.....
Composant #N <ident>
8'
4
Dj
l2
Composant #1 1.1.5.3
Composant #2 5.4.2.0
Composant #3 25/06/96
Composant #4 32.104
Composant #5 12
.....
Composant #N <ident>
Composant #3 28/11/97
Composant #5 15
7'

FIG.5B

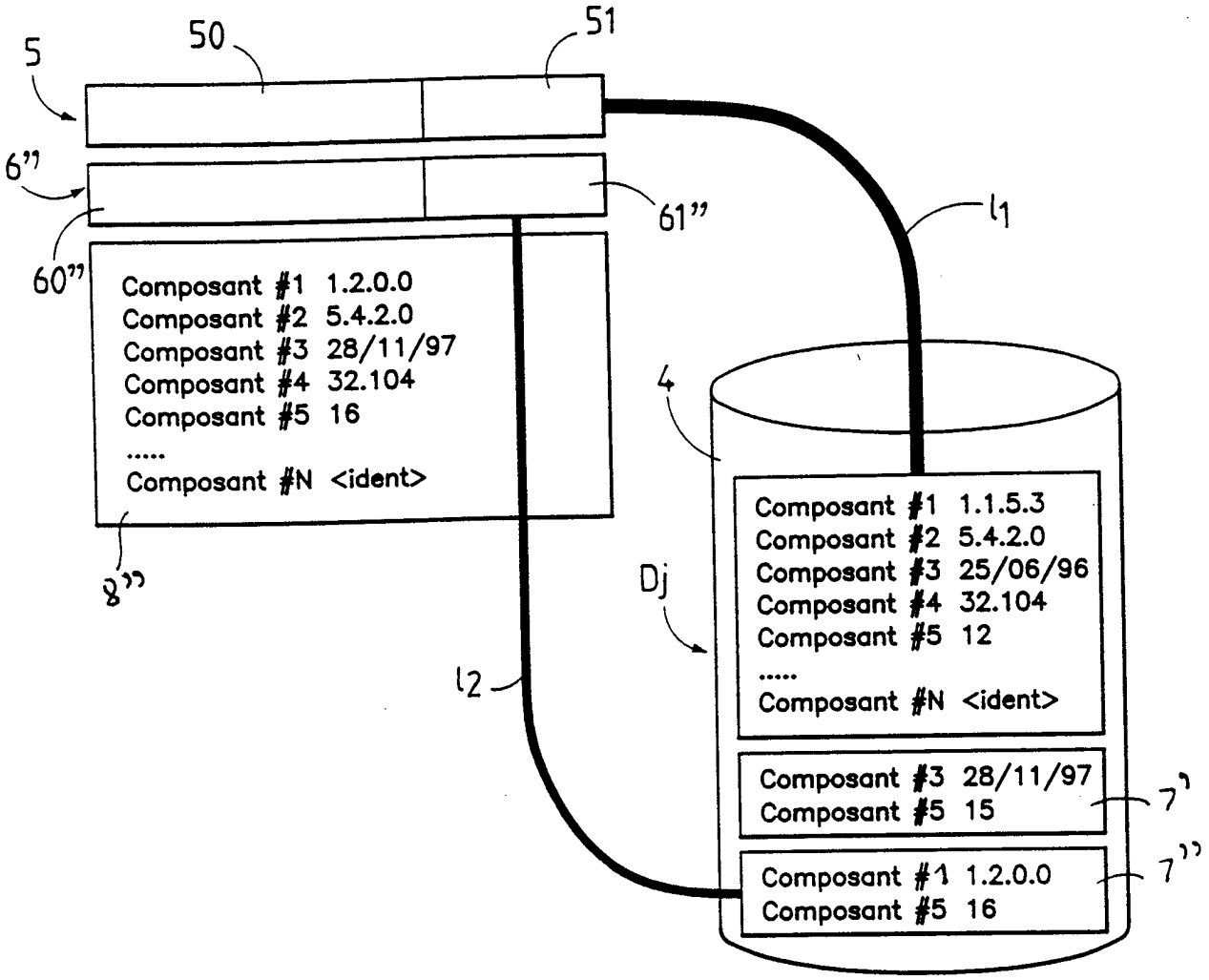

5
50
51
6"
61"
l1
60"
Composant #1 1.2.0.0
Composant #2 5.4.2.0
Composant #3 28/11/97
Composant #4 32.104
Composant #5 16
.....
Composant #N <ident>
8"
4
Dj
l2
Composant #1 1.1.5.3
Composant #2 5.4.2.0
Composant #3 25/06/96
Composant #4 32.104
Composant #5 12
.....
Composant #N <ident>
Composant #3 28/11/97
Composant #5 15
7'
Composant #1 1.2.0.0
Composant #5 16
7"

FIG.5C

INTERNATIONAL SEARCH REPORT

Inter ɔnal Application No
PCT/FR 98/02887

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 528 617 A (SUN MICROSYSTEMS INC) 24 February 1993	1-4,8
Y	see column 2, line 26 - line 41 see column 7, line 47 - column 10, line 33 see column 16, line 4 - line 16	5-7,9-11
Y	US 5 649 200 A (LEBLANG DAVID B ET AL) 15 July 1997 see page 26, line 9 - page 28, line 36	5-7,9-11

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
9 June 1999	16/06/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Kingma, Y

1

INTERNATIONAL SEARCH REPORT

..formation on patent family members

Inter ...nal Application No
PCT/FR 98/02887

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0528617 A	24-02-1993	JP 7044373 A	14-02-1995
		US 5481722 A	02-01-1996
US 5649200 A	15-07-1997	NONE	

RAPPORT DE RECHERCHE INTERNATIONALE

Dem : Internationale No
PCT/FR 98/02887

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
CIB 6 G06F9/44

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
CIB 6 G06F

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si réalisable, termes de recherche utilisés)

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie °	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X Y	EP 0 528 617 A (SUN MICROSYSTEMS INC) 24 février 1993 voir colonne 2, ligne 26 - ligne 41 voir colonne 7, ligne 47 - colonne 10, ligne 33 voir colonne 16, ligne 4 - ligne 16	1-4,8 5-7,9-11
Y	US 5 649 200 A (LEBLANG DAVID B ET AL) 15 juillet 1997 voir page 26, ligne 9 - page 28, ligne 36	5-7,9-11

☐ Voir la suite du cadre C pour la fin de la liste des documents

☒ Les documents de familles de brevets sont indiqués en annexe

° Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'inven tion revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'inven tion revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
9 juin 1999	16/06/1999

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Kingma, Y

1

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Dem : Internationale No
PCT/FR 98/02887

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
EP 0528617 A	24-02-1993	JP 7044373 A	14-02-1995
		US 5481722 A	02-01-1996
US 5649200 A	15-07-1997	AUCUN	