(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 816 080**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **00 13917**

(51) Int Cl[7] : **G 06 F 17/00**

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 30.10.00.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 03.05.02 Bulletin 02/18.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Se reporter à la fin du présent fascicule*

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *FRANCE TELECOM Société anonyme* — FR.

(72) **Inventeur(s) :** MALVILLE ERIC et MILHAU MICHEL.

(73) **Titulaire(s) :**

(74) **Mandataire(s) :** FRANCE TELECOM.

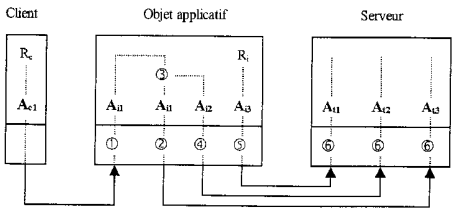
(54) PROCEDE DE PROPAGATION DE CONTEXTES D'INVOCATION A TRAVERS UN SYSTEME DISTRIBUE A OBJETS.

(57) Procédé de propagation de contextes d'invocation à travers un système distribué à objets, ledit système comportant une interface API de gestion d'activités et chaque objet étant associé à un intercepteur et à un authentificateur de contexte de requête entrante reçue par l'intercepteur lors d'une invocation dudit objet.

Selon l'invention, ledit procédé comprend les opérations consistant à:

- créer un gestionnaire d'activités destiné à l'intercepteur pour, notamment, d'une part, établir une association entre une activité d'un objet consécutive directement à une requête entrante invoquant ledit objet et le contexte de ladite requête entrante tel qu'identifié par l'authentificateur de contexte, et, d'autre part, récupérer le contexte dudit gestionnaire d'activités et l'associer à la requête sortante de l'objet,
- ajouter à ladite interface API de gestion d'activités une méthode destinée à récupérer le contexte courant du gestionnaire d'activités et de l'associer à toute activité de l'objet consécutive indirectement à la requête entrante.

Application à la sécurité des systèmes distribués à objets.



FR 2 816 080 - A1

La présente invention concerne un procédé de propagation de contextes d'invocation à travers un système distribué à objets.

L'invention trouve une application avantageuse dans tous les cas où il est nécessaire de connaître, notamment par leur contexte, l'ordre causal des interactions entre les objets, c'est-à-dire, de connaître pour chaque invocation, quelles sont les invocations qu'elle a engendrées. On sait en effet que les services offerts par ce type de système distribué à objets sont réalisés par des interactions entre les objets qui le composent, l'invocation d'un service pouvant ainsi engendrer une succession d'appels à des méthodes de différents objets.

L'invention s'applique plus particulièrement à la sécurité des systèmes distribués à objets. Il est fréquent que l'accès à une méthode d'un objet soit conditionné par l'identité de l'initiateur, le plus souvent l'utilisateur. Pour contrôler l'accès aux méthodes des objets, il est donc nécessaire de fournir l'identité du client à tous les objets participant à la réalisation du service demandé.

Dans d'autres applications, comme la supervision ou l'audit, il est également nécessaire de connaître l'identité, et, plus généralement, le contexte des différents intermédiaires ayant participé à la réalisation du service.

Le standard CORBA (*Common Object Request Broker Architecture*) de l'OMG (*Object Management Group*) propose un certain nombre de spécifications destinées au développement d'infrastructures de communication orientées objets. Parmi ces spécifications, on trouve celle des intercepteurs. Les intercepteurs s'intercalent entre l'ORB (*Object Request Broker*), le noyau de l'infrastructure de communication, et les objets applicatifs, et permettent de réaliser des traitements sur les requêtes entrantes ou sortantes de manière transparente pour les objets applicatifs. Ils sont donc naturellement utilisés pour implémenter des services que l'on veut transparents pour les objets applicatifs :audit, contrôle d'accès, supervision, etc.

Un certain nombre d'implémentations de ces spécifications CORBA propose un service d'intercepteurs. En revanche, aucun de ces produits ne

permet la propagation transparente d'informations; la propagation ne pouvant se faire sans que les objets applicatifs ne participent explicitement. Il devient alors impossible dans certains cas d'implémenter des services totalement transparents pour les objets applicatifs.

Les intercepteurs permettent donc de réaliser des actions sur les requêtes entrantes ou sortantes de façon transparente pour l'objet CORBA. La plupart des ORB implémentant cette notion d'intercepteur n'offrent pas, en revanche, de mécanisme de propagation transparente de l'information. Le problème réside dans le fait qu'un intercepteur n'a aucun moyen de savoir, dans tous les cas, qu'elle est la relation entre les requêtes entrantes et les requêtes sortantes. Pour offrir un mécanisme de propagation transparent, il est nécessaire pour un intercepteur de savoir, pour chaque requête sortante, si elle est consécutive à une requête entrante, et si oui laquelle, ou s'il s'agit d'une requête envoyée spontanément par l'objet.

Comme on peut le voir sur la figure 1 qui illustre le mécanisme d'invocation d'un objet applicatif dans un système à intercepteur connu, lorsqu'un objet applicatif reçoit une invocation une nouvelle activité A_1 pour le traitement de cette invocation est créée au moyen d'une interface de programmation applicative (API pour « Application Programming Interface ») de gestion d'activités. Un authentificateur identifie le contexte de l'invocation et donc l'activité créée. Si l'objet utilise cette même activité A_1 pour réaliser une invocation d'un autre objet, l'intercepteur peut directement vérifier qu'il s'agit d'une requête sortante consécutive à la requête entrante et lui associer le contexte identifié par l'authentificateur. Seul ce cas de figure ne soulève aucune ambiguïté. En effet, si la requête reçue engendre la création d'une nouvelle activité A_2 dédiée à l'invocation d'un autre objet, l'intercepteur n'est plus en mesure de savoir si cette invocation est consécutive à la requête entrante ou s'il s'agit d'une requête spontanée A_3. Il est donc impossible de propager le contexte relatif aux requêtes entrantes des invocations de l'objet.

Dans ce cas, la propagation du contexte nécessite une modification du code des objets applicatifs. Il est en effet nécessaire de demander explicitement à l'application d'associer le contexte de la requête entrante à la nouvelle activité créée A_2. Or, les codes des objets applicatifs ne sont généralement pas disponibles pour que les développeurs puissent effectuer cette modification.

Les services d'infrastructure offerts par les systèmes distribués à objets actuels ne permettent donc pas de déterminer de manière transparente l'ordre causal des invocations. La présente invention est destinée à enrichir les services offerts par ces infrastructures pour garantir cette propriété de traçabilité transparente.

Aussi, le problème technique à résoudre par l'objet de la présente invention est de proposer un procédé de propagation de contextes d'invocation à travers un système distribué à objets, ledit système comportant une interface API de gestion d'activités et chaque objet étant associé à un intercepteur et à un authentificateur de contexte de requête entrante reçue par l'intercepteur lors d'une invocation dudit objet, procédé qui permettrait de déterminer l'ordre causal des invocations de manière transparente et donc, de connaître l'identité des objets impliqués dans la réalisation d'un service, ainsi que l'ordre dans lequel ils sont activés, sans que les objets eux-mêmes n'aient à intervenir directement dans le processus, offrant donc notamment la possibilité de mettre en place une politique de contrôle d'accès fine.

La solution au problème technique posé consiste, selon, la présente invention, en ce que ledit procédé comprend les opérations consistant à :

- créer un gestionnaire d'activités destiné à l'intercepteur pour, notamment, d'une part, établir une association entre une activité d'un objet consécutive directement à une requête entrante invoquant ledit objet et le contexte de ladite requête entrante tel qu'identifié par l'authentificateur de contexte, et, d'autre part, récupérer le contexte dudit gestionnaire d'activités et l'associer à la requête sortante de l'objet,
- ajouter à ladite interface API de gestion d'activités une méthode destinée à récupérer le contexte courant du gestionnaire d'activités et de l'associer à toute activité de l'objet consécutive indirectement à la requête entrante.

Ainsi, en créant un gestionnaire d'activités et en enrichissant l'API offerte par le langage de développement pour la gestion d'activités (création, synchronisation, destruction, ...), le procédé de l'invention permet de maintenir à jour les associations entre tous les types d' activités et les contextes d'invocation. Ces associations peuvent ensuite être utilisées par les intercepteurs pour récupérer les contextes associés aux requêtes sortantes.

La propagation des contextes ne nécessite alors plus l'intervention des objets applicatifs, ce qui permet le développement de services totalement transparents.

Le procédé selon l'invention fonctionne de la manière suivante. Lorsque l'intercepteur reçoit une requête, il en extrait le contexte et demande au gestionnaire d'activités de créer une association entre ce contexte et l'activité courante consécutive directement à la requête entrante. L'invocation est ensuite transmise à l'objet applicatif.

Si cette même activité est utilisée par l'objet applicatif pour invoquer un autre objet, l'intercepteur peut directement récupérer le contexte relatif à l'activité courante en interrogeant le gestionnaire d'activités et l'associer à la requête sortante.

Lorsque l'objet applicatif désire crée une nouvelle activité pour invoquer un autre objet, il doit utiliser l'API de gestion d'activités enrichie. La nouvelle activité créée par cette API, consécutive indirectement à la requête entrante, hérite du contexte de l'activité mère. Il suffit alors à l'intercepteur de récupérer directement le contexte de l'activité courante , à savoir l'activité fille, et l'associer à la requête sortante.

Si l'activité courante n'est pas consécutive, directement ou indirectement, à une requête entrante, aucun contexte ne lui est associé. C'est le cas notamment lorsque l'objet applicatif invoque spontanément un autre objet. Dans ce cas, l'activité n'a pas été créée pour gérer une requête entrante. Elle n'est pas non plus la fille d'une activité dédiée à la gestion d'une requête entrante. Elle n'est donc associée à aucun contexte.

La description qui va suivre en regard des dessins annexés, donnés à titre d'exemples non limitatifs, fera bien comprendre en quoi consiste l'invention et comment elle peut être réalisée.

La figure 2 est un schéma d'un système distribué à objets fonctionnant selon le procédé conforme à l'invention.

La figure 3 est une variante du système distribué de la figure 2.

On se placera dans le cas où le besoin exprimé pour sécuriser le système distribué à objets des figures 2 et 3 est de pouvoir propager de manière transparente jusqu'au serveur et à travers l'objet applicatif le contexte particulier consistant en l'identité du client, initiateur des invocations.

De manière connu, le système distribué représenté aux figures 2 et 3 comporte une interface API de gestion d'activités définie par la classe Thread standard comprenant la méthode Thread().

Les objets (client, applicatif et serveur) comportent des intercepteurs aptes à modifier les requêtes pour y ajouter et extraire un contexte d'invocation, notamment la liste des identités des objets impliqués dans la chaîne d'invocation.

Par ailleurs, il est prévu que les intercepteurs comprennent un mécanisme d'authentification, par exemple SSL, entre les objets. Ce mécanisme est classiquement utilisé par les intercepteurs pour permettre à un objet serveur d'authentifier le contexte d'une requête entrante invoquée par un objet client.

De manière à pouvoir propager les contextes d'invocation à travers le système distribué, il est créé un gestionnaire d'activités ThreadManager offrant aux intercepteurs une API leur permettant :

- d'ajouter une nouvelle association (méthode setContext) entre l'activité courante et le nouveau contexte correspondant à la requête entrante (paramètre context).
- de récupérer, s'il existe, le contexte associé à l'activité courante (méthode getContext).
- de supprimer l'association entre l'activité courante et son contexte (méthode deleteContext).

void	**setContext**(Object context)
Object	**getContext**()
void	**deleteContext**()

Cet objet est une classe à instance unique (et donc partagé par tous les objets du même processus, et donc en particulier par les intercepteurs) qu'il est possible de récupérer de la manière suivante :

```
private static ThreadManager manager = ThreadManager.getInstance();
```

De même, la propagation transparente des contextes nécessite d'enrichir l'API de gestion des activités afin d'associer le contexte courant à une nouvelle activité créée par l'objet applicatif.

Deux méthodes sont possibles pour enrichir une API : modifier l'API offerte par le langage de programmation ou développer une API spécifique héritant de l'API standard et enrichissant les méthodes qu'elle propose. La première approche offre un niveau de transparence maximal. En revanche, il est nécessaire dans ce cas de disposer des sources du langage de développement, ce qui n'est pas toujours possible. L'exemple présenté ici consiste à développer une API spécifique selon la deuxième approche.

Une classe FtThread permet d'associer un contexte à chacune de ses instances. Cette classe hérite de la classe Thread standard et offre les mêmes constructeurs (i.e. les mêmes signatures). Une méthode supplémentaire getContext permet de récupérer le contexte associé à une activité.

	FtThread()
	FtThread(Runnable target)
	FtThread(Runnable target,String name)
	FtThread(String name)
	FtThread(ThreadGroup group,Runnable target)
	FtThread(ThreadGroup group,Runnable target,String name)
	FtThread(ThreadGroup group,String name)
Object	**getContext**()

Pour permettre une propagation d'informations, les objets applicatifs doivent utiliser cette classe pour la création d'activités.

En Java par exemple, deux méthodes sont possibles pour créer une activité. La première consiste à créer une classe héritant de la classe FtThread.

```
public class ExtendThread extends FtThread {
        ExtendThread() { ... }
        public void run() { ... }
}
```

La création et l'exécution de l'activité sont alors réalisées de la manière suivante :

```
ExtendThread thread = new ExtendThread(target);
thread.start();
```

La seconde méthode consiste à créer une classe implémentant l'interface Runnable.

```
public class ImplementThread implements Runnable {
        ImplementThread() { ... }
        public void run() { ... }
}
```

La création et l'exécution de l'activité sont alors réalisées de la manière suivante :

```
ImplementThread thread = new ImplementThread();
new FtThread(thread).start();
```

Le procédé de propagation de contextes conforme à l'invention fonctionne selon les opérations suivantes représentées sur la figure 2 :

1. Lorsque l'intercepteur de l'objet applicatif reçoit une requête, il authentifie le client, crée un contexte contenant l'identité du client et associe ce contexte à l'activité courante A_{i1}, consécutive directement à la requête entrante, en utilisant le gestionnaire d'activités ThreadManager créé.
2. Si l'objet applicatif utilise cette même activité A_{i1} pour invoquer le serveur, l'intercepteur récupère le contexte associé à cette activité en utilisant le gestionnaire d'activités ThreadManager et insère l'identité du client dans la requête envoyée.
3. Pour créer une nouvelle activité A_{i2}, consécutive indirectement à la requête entrante, et l'utiliser pour invoquer le serveur, l'objet applicatif doit utiliser l'API de gestion d'activités enrichie FtThread. La nouvelle activité A_{i2} ainsi créée est alors associée de manière transparente au contexte de l'activité A_{i1}.

4. Le gestionnaire d'activités ThreadManager permet à l'intercepteur de récupérer le contexte associé à l'activité A_{i2} et d'ajouter l'identité du client à la requête sortante.
5. Lorsque l'objet applicatif invoque le serveur de manière spontanée, c'est à dire. en utilisant une activité A_{i3} qui n'a pas été engendrée par la réception d'une requête, l'intercepteur peut alors vérifier que l'activité courante A_{i3} n'est associée à aucun contexte. L'identité du client n'est donc pas ajoutée à la requête sortante.
6. A la réception d'une requête, l'intercepteur du serveur authentifie l'objet applicatif et extrait le contexte associé à la requête. Il peut alors déterminer quel est le chemin suivi par la requête reçue.

Le serveur peut également utiliser le gestionnaire d'activités ThreadManager pour récupérer le contexte associé à l'activité courante et donc déterminer le chemin parcouru par la requête.

La figure 3 illustre une variante du procédé de l'invention dans laquelle l'objet applicatif stocke les requêtes entrantes dans une file de messages et utilise un ensemble (i.e. un "*pool*") d'activités destiné à transmettre les requêtes reçues aux destinataires appropriés. Les API offertes ne suffisent plus pour permettre une propagation transparente d'informations. Ces activités ne sont, en effet, associées à aucun contexte.

En revanche, elles permettent aux développeurs de services de sécurité d'offrir des API permettant de gérer ce type de cas spécifique. Dans l'exemple proposé en regard de la figure 2, il suffit aux développeurs de créer une API de gestion d'une file de messages offrant notamment deux méthodes (put et get) permettant respectivement de déposer et de récupérer un message dans la file. La méthode put permet d'associer le contexte de l'activité courante au message déposé dans la file. La méthode get permet d'associer le contexte du message retiré à l'activité courante.

La variante de la figure 3 fonctionne de la manière suivante :

1. L'intercepteur détermine le contexte de la requête entrante et l'associe à l'activité courante A_1 en utilisant le gestionnaire d'activités ThreadManager.
2. L'objet applicatif utilise la méthode put pour insérer le message dans la file de messages. Le contexte de l'activité courante A_1 est associé de manière transparente au message.

3. L'objet applicatif retire un message de la file de messages en utilisant la méthode get. Le contexte du message est associé à l'activité courante A_2.
4. L'intercepteur récupère le contexte de l'activité courante A_2.

REVENDICATIONS

1-Procédé de propagation de contextes d'invocation à travers un système distribué à objets, ledit système comportant une interface API de gestion d'activités et chaque objet étant associé à un intercepteur et à un authentificateur de contexte de requête entrante reçue par l'intercepteur lors d'une invocation dudit objet, caractérisé en ce que ledit procédé comprend les opérations consistant à :

- créer un gestionnaire d'activités destiné à l'intercepteur pour, notamment, d'une part, établir une association entre une activité d'un objet consécutive directement à une requête entrante invoquant ledit objet et le contexte de ladite requête entrante tel qu'identifié par l'authentificateur de contexte, et, d'autre part, récupérer le contexte dudit gestionnaire d'activités et l'associer à la requête sortante de l'objet,
- ajouter à ladite interface API de gestion d'activités une méthode destinée à récupérer le contexte courant du gestionnaire d'activités et de l'associer à toute activité de l'objet consécutive indirectement à la requête entrante.

2-Procédé selon la revendication 1, caractérisé en ce qu'il comprend également une opération consistant à créer une interface API de gestion d'une file de messages, offrant deux méthodes permettant respectivement de déposer un message dans la file et d'associer le contexte de l'activité courante au message déposé dans la file, et de récupérer un message dans la file et d'associer le contexte du message retiré à l'activité courante.

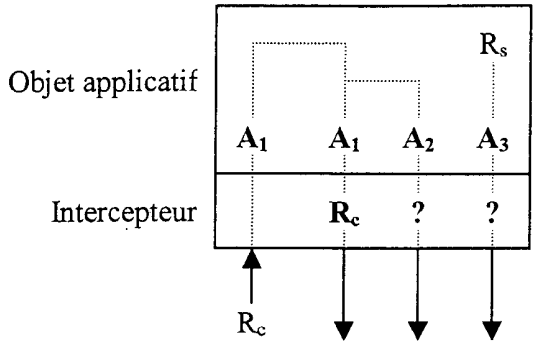


FIG. 1 Art antérieur

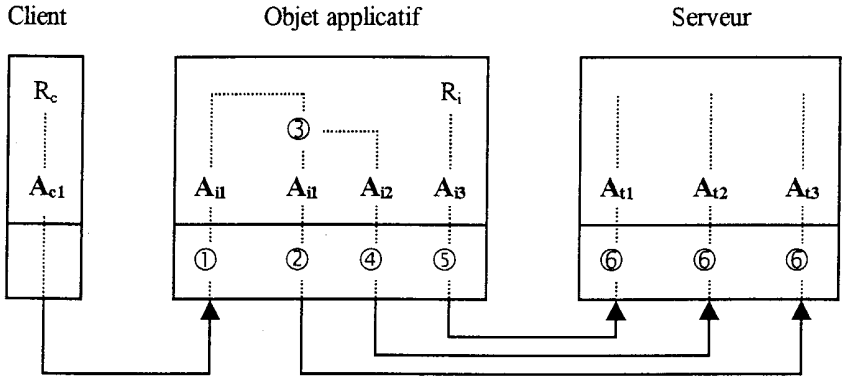


FIG. 2

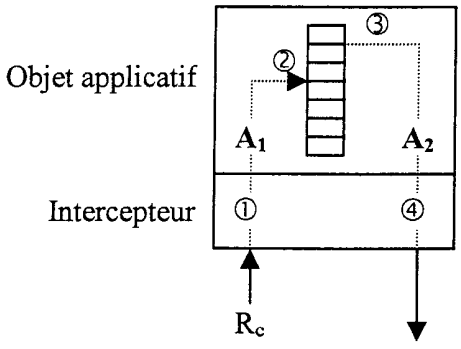


FIG. 3

RÉPUBLIQUE FRANÇAISE

2816080

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

N° d'enregistrement national

FA 596905
FR 0013917

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
A	NARASIMHAN P., MOSER L. E., MELLIAR-SMITH P. M.: "USING INTERCEPTORS TO ENHANCE CORBA" COMPUTER, IEEE COMPUTER SOCIETY PRESS, US, 'en ligne! vol. 32, no. 7, juillet 1999 (1999-07), pages 62-68, XP002171364 ISSN: 0018-9162 Extrait de l'Internet: <URL:http://ieeexplore.ieee.org/iel5/2/16817/00774920.pdf> 'extrait le 2001-07-06! * page 63, colonne de gauche, ligne 48 - colonne de droite, ligne 29 * * page 65, colonne de droite, ligne 15 - colonne de gauche, ligne 14 * page 65, allinéa "Security" ---	1,2	G06F17/00
A	WO 99 28819 A (EDWARDS NIGEL JOHN ;REES OWEN RICHARD THOMAS (GB); HEWLETT PACKARD) 10 juin 1999 (1999-06-10) * abrégé * * page 2, ligne 12 - page 3, ligne 29 * * page 5, ligne 17 - ligne 33 * * page 7, ligne 28 - ligne 30 * * page 8, ligne 23 - page 9, ligne 2 * * page 10, ligne 26 - page 11, ligne 22 * * page 12, ligne 16 - page 14, ligne 10 * * page 17, ligne 19 - ligne 25 * --- -/--	1,2	

DOMAINES TECHNIQUES RECHERCHÉS (Int.CL.7)

G06F

1

Date d'achèvement de la recherche	Examinateur
10 juillet 2001	Archontopoulos, E

CATÉGORIE DES DOCUMENTS CITÉS

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : arrière-plan technologique
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

& : membre de la même famille, document correspondant

EPO FORM 1503 12.99 (P04C14)

RÉPUBLIQUE FRANÇAISE

2816080

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

N° d'enregistrement national

FA 596905
FR 0013917

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
A	OBJECT MANAGEMENT GROUP, INC.: "THE COMMON OBJECT REQUEST BROKER: ARCHITECTURE AND SPECIFICATION. REVISION 2.3" OMG, 'en ligne! juin 1999 (1999-06), pages 21-1--21-9, XP002171365 Extrait de l'Internet: <URL:http://cgi.omg.org/cgi-bin/doc?formal/98-12-01.pdf> 'extrait le 2001-07-06! * alinéa '21.1! - alinéa '21.2! * * alinéa '21.2.2! * * alinéa '21.4.1! * ---	1,2	
A	US 5 920 863 A (MCKEEHAN MICHAEL DENNIS ET AL) 6 juillet 1999 (1999-07-06) * abrégé * * colonne 3, ligne 60 - colonne 4, ligne 25 * * colonne 10, ligne 7 - ligne 63 * ---	1,2	
A	PATENT ABSTRACTS OF JAPAN vol. 2000, no. 04, 31 août 2000 (2000-08-31) -& JP 2000 020329 A (HITACHI LTD), 21 janvier 2000 (2000-01-21) * abrégé * -----	1,2	DOMAINES TECHNIQUES RECHERCHÉS (Int.CL.7)

1

Date d'achèvement de la recherche	Examinateur
10 juillet 2001	Archontopoulos, E

CATÉGORIE DES DOCUMENTS CITÉS

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : arrière-plan technologique
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

& : membre de la même famille, document correspondant

EPO FORM 1503 12.99 (P04C14)