(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 913 275**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **07 53632**

(51) Int Cl[8] : ***G 06 F 17/22*** *(2006.01), H 03 M 7/00*

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 02.03.07.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 05.09.08 Bulletin 08/36.

(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) :** *CANON KABUSHIKI KAISHA* — JP.

(72) **Inventeur(s) :** RUELLAN HERVE.

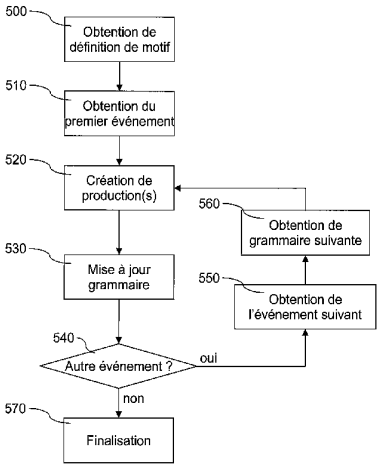
(73) **Titulaire(s) :**

(74) **Mandataire(s) :** SANTARELLI.

(54) PROCEDE ET DISPOSITIF DE CODAGE D'UN DOCUMENT ET PROCEDE ET DISPOSITIF DE DECODAGE D'UN DOCUMENT.

(57) Le procédé de codage de données hiérarchisées organisées en une pluralité d'items pouvant êtres décrits par un ensemble d'événements en utilisant au moins une grammaire comportant des productions, ledit procédé comportant une étape d'adjonction d'au moins une production à une dite grammaire, étape d'adjonction qui comporte:

- une étape d'obtention d'un motif structurel représentant une pluralité d'informations au moins structurelles représentatives d'au moins un item à coder;
- une étape (510 à 560) d'extraction d'une pluralité d'événements décrivant ledit motif structurel obtenu;
- une étape (520) de création d'au moins une production comportant, ensemble, la pluralité d'événements extraits; et
- une étape (530) d'insertion de chaque dite production créée dans une dite grammaire.



FR 2 913 275 - A1

La présente invention concerne un procédé et un dispositif de codage de document et un procédé et un dispositif de décodage de document. Elle s'applique, en particulier, au langage XML (acronyme de « Extensible Markup Language » pour langage de balisage extensible). Ce langage est une syntaxe pour définir des langages informatiques. XML permet ainsi de créer des langages adaptés à des utilisations différentes mais pouvant être traités par les mêmes outils.

Un document XML est composé d'éléments, chaque élément commençant par une balise ouvrante comportant le nom de l'élément (par exemple: <balise>) et se terminant par une balise fermante comportant, elle aussi, le nom de l'élément (par exemple: </balise>). Chaque élément peut contenir d'autres éléments, appelés « éléments enfants » (une terminologie de filiation, « parent », « enfant », étant utilisée pour décrire les relations entre les éléments imbriqués) ou des données textuelles.

D'autre part, un élément peut être précisé par des attributs, chaque attribut étant défini par un nom et ayant une valeur. Les attributs sont placés dans la balise ouvrante de l'élément qu'ils précisent (par exemple: <balise attribut="valeur">).

La syntaxe XML permet aussi de définir des commentaires (par exemple: <!-- Commentaire-->) et des instructions de traitement, qui peuvent préciser à une application informatique les traitements à appliquer au document XML (par exemple: « <?montraitement?> »).

Dans la terminologie XML, l'ensemble des termes « élément », « attribut », « donnée textuelle », « commentaire » et « instruction de traitement » sont regroupés sous le nom générique de « item ». Dans un cadre plus général, l'ensemble de ces termes peut être regroupé sous le nom générique de « nœud ».

Plusieurs langages XML différents peuvent contenir des éléments de même nom. Pour pouvoir mélanger plusieurs langages XML différents, un ajout a été effectué à la syntaxe XML permettant de définir des espaces de nommage (« Namespace » en anglais). Deux éléments seront identiques seulement s'ils ont le même nom et se trouvent dans le même espace de nommage. Un espace de nommage est défini par une URI (acronyme de « Uniform Resource Identifier », pour Identifiant uniforme de ressource), par exemple « http://canon.crf.fr/xml/monlangage ». L'utilisation d'un espace de nommage dans un document XML passe par la définition d'un préfixe qui est un raccourci vers l'URI de cet espace de nommage. Ce préfixe est défini à l'aide d'un attribut spécifique (par exemple: « xmlns:ml="http://canon.crf.fr/xml/monlangage" » associe le préfixe « ml » à l'URI « http://canon.crf.fr/xml/monlangage »). Ensuite, l'espace de nommage d'un élément ou d'un attribut sera précisé en faisant précéder son nom par le préfixe associé à l'espace de nommage suivi de « : » (par exemple: « <ml:balise ml:attribut="valeur"> »).

Le standard XML Schema définit un langage permettant de décrire la structure d'un ensemble de documents XML. Un document XML Schema est un document XML, et décrit l'ensemble des éléments et des attributs pouvant être présents dans un document XML conforme à ce document XML Schema, ainsi que les relations entre ces éléments et ces attributs.

D'autres systèmes permettent de décrire la structure d'un ensemble de documents XML, comme les DTD (acronyme de « Document Type Definition », pour Définition de Type de Document) ou comme le langage Relax NG.

XML présente de nombreux avantages et est devenu un standard pour stocker des données dans un fichier ou pour échanger des données. XML permet en particulier de disposer de nombreux outils pour traiter les fichiers générés. D'autre part, un document XML peut être édité manuellement avec un simple éditeur de texte. En outre, comme un document XML contient sa structure intégrée aux données, ce document est très lisible même sans en connaître la spécification.

Le principal inconvénient de la syntaxe XML est d'être très prolixe. Ainsi la taille d'un document XML peut être plusieurs fois supérieure à la taille intrinsèque des données. Cette taille importante des documents XML induit aussi un temps de traitement important lors de la génération et de la lecture de documents XML. Elle induit aussi un temps de transmission important.

Pour remédier à ces inconvénients, d'autres méthodes pour encoder un document XML ont été recherchées. Le but de ces méthodes est de coder le contenu du document XML sous une forme plus efficace, mais permettant de reconstruire facilement le document XML. Cependant, la plupart de ces méthodes ne conservent pas l'ensemble des avantages du format XML.

Parmi ces méthodes, la plus simple consiste à coder les données de structure dans un format binaire au lieu d'utiliser un format textuel. En outre, la redondance des informations structurelles dans le format XML peut être supprimée ou au moins diminuée (par exemple, il n'est pas forcément utile de préciser le nom de l'élément dans la balise ouvrante et la balise fermante).

Une autre méthode est d'utiliser une table d'index, en particulier pour les noms d'éléments et d'attributs qui sont généralement répétés dans un document XML. Ainsi, lors de la première occurrence d'un nom d'élément, celui-ci est codé normalement dans le fichier et un index lui est associé. Puis, pour les occurrences suivantes de ce nom d'élément, l'index sera utilisé à la place de la chaîne complète, réduisant la taille du document généré, mais facilitant aussi la lecture (il n'y a plus besoin de lire la chaîne complète dans le fichier, et en outre, la détermination de l'élément lu peut être réalisée par une comparaison d'entiers au lieu d'une comparaison de chaînes de caractères).

La spécification « Efficient XML » est à la base du futur standard XML Binaire du W3C (acronyme de « World Wide Web Consortium », organisation produisant des standards pour le Web). Cette spécification utilise un ensemble de grammaires pour coder un document XML.

Pour pouvoir coder les items compris dans un document XML, la spécification Efficient XML divise chacun de ces items en parties élémentaires appelées événements. Ces événements sont similaires à ceux générés par des parseurs XML travaillant en mode streaming, c'est-à-dire représentant un

document XML comme un flux de données, tels que les parseurs SAX (acronyme de « Simple API for XML », pour Interface de programmation simple pour XML). Ainsi, par exemple, dans la spécification Efficient XML, un élément XML sera représenté par un événement de début d'élément, un ensemble d'événements représentant ses attributs, un autre ensemble d'événements représentant son contenu et un événement de fin d'élément.

Une grammaire est composée d'un ensemble de productions, chaque production comprenant une description d'événement XML, une valeur de codage associée et l'indication de la grammaire suivante à utiliser. Pour coder un événement XML à l'aide d'une grammaire, la production contenant la description la plus précise de l'événement XML est utilisée. La valeur de codage contenue dans cette production est utilisée pour représenter l'événement, et les informations contenues dans l'événement et non décrites dans la production sont codées.

Une grammaire est évolutive. Dans un certain nombre de cas, après l'occurrence d'un événement XML déjà décrit par une production de la grammaire (s'il n'est pas décrit par une production, il ne peut être encodé par la grammaire), la grammaire est modifiée pour inclure une nouvelle production correspondant à cet événement XML. Cette production peut soit contenir une description plus précise de l'événement, diminuant le nombre d'informations à coder pour représenter l'événement, soit avoir une valeur de codage plus compacte.

Les valeurs de codage, ou « codes », sont exprimées sous forme de « priorités » ayant entre 1 et 3 niveaux. Coder une valeur de codage revient à coder les valeurs de sa priorité. Chaque niveau est codé sur le nombre de bits minimum pour pouvoir coder la plus grande valeur de ce niveau associée à une production de la grammaire.

Pour coder un document XML, un ensemble de grammaires est utilisé. Quelques grammaires servent à coder la structure propre au document XML. En outre, pour chaque type d'élément XML présent dans le document (un type d'élément XML étant un ensemble d'éléments ayant le même nom), un ensemble de grammaires est utilisé pour coder les éléments XML de ce type.

Les règles de grammaires utilisées peuvent soit être des règles génériques, communes à tous les documents XML et construites à partir de la syntaxe XML, soit être des règles spécifiques à un type de document, construites à partir d'un Schéma XML décrivant la structure de ce type de document.

Lors du décodage, le processus inverse est utilisé : la valeur de codage est extraite et permet d'identifier l'événement XML codé, ainsi que les informations complémentaires à décoder.

En outre, lors du décodage, les mêmes règles d'évolution des grammaires sont utilisées, permettant d'avoir à tout moment un ensemble de règles de grammaires identique à celui qui était utilisé lors du codage.

A titre d'exemple, le fragment XML suivant est utilisé pour décrire le codage d'un document XML à l'aide de la spécification Efficient XML.

```
<person>
   <firstname>John</firstname>
   <lastname>Smith</lastname>
</person>
```

La grammaire utilisée pour coder le contenu de l'élément « person » est la suivante (de manière simplifiée par rapport à la spécification Efficient XML) :

ElementContent :	
EE	0
SE (*) ElementContent	1.0
CH ElementContent	1.1

« EE » correspond à l'événement de fin d'élément, « SE (*) » correspond un événement de début d'élément quelconque (le nom n'est pas précisé), et « CH » correspond à un événement de contenu textuel.

Lors de l'encodage, après avoir reçu l'événement correspondant au début d'élément « person », « SE (person) », et l'avoir codé, l'encodeur sélectionne la grammaire de codage du contenu de l'élément « person », décrite ci-dessus.

Ensuite, l'encodeur reçoit l'événement correspondant au début d'élément « firstname », « SE (firstname) ». La production qui correspond à cet événement dans la grammaire ci-dessus est la deuxième :

SE (*) ElementContent 1.0

L'encodeur va donc coder la priorité « 1.0 ». Comme le premier niveau de priorité comprend deux valeurs distinctes (« 0 » et « 1 »), ce niveau peut être codé sur un bit, avec la valeur « 1 ». De même, le deuxième niveau de priorité comprend deux valeurs distinctes et peut être codé sur un bit, avec la valeur « 0 ». La priorité « 1.0 » est donc codée ici avec les deux bits « 10 ».

Ensuite, comme la production ne précise pas le nom de l'élément, « firstname » est codé.

Ensuite, l'encodeur va coder le contenu de « firstname » (ce codage est réalisé à partir de la grammaire associée à l'élément « firstname », de manière similaire).

Quand le contenu de « firstname » est codé, l'encodeur modifie la grammaire associée à l'élément « person » pour adapter la grammaire. Pour cela, une nouvelle production est ajoutée à la grammaire, cette production correspondant au début d'élément « firstname ». A cette production est associée la priorité « 0 », et les autres priorités sont décalées pour conserver l'unicité des priorités. Ainsi la grammaire devient :

ElementContent :

SE (firstname) ElementContent	0
EE	1
SE (*) ElementContent	2.0
CH ElementContent	2.1

L'événement suivant est le début de l'élément « lastname ». Comme pour « firstname », cet élément est codé à l'aide de la production :

SE (*) ElementContent 2.0

Le premier niveau de priorité ayant maintenant trois valeurs possibles, il est codé sur 2 bits, avec la valeur « 2 ». Le deuxième niveau de priorité est toujours codé sur un seul bit. La priorité « 2.0 » est donc codée ici avec les trois bits « 100 ».

Le nom de l'élément, « lastname » est ensuite codé. Puis le contenu de « lastname » est codé à l'aide de la grammaire associée à l'élément « lastname ».

Ensuite, la grammaire est modifiée pour y ajouter une production correspondant au début de l'élément « lastname » et elle devient alors :

ElementContent :

SE (lastname) ElementContent	0
SE (firstname) ElementContent	1
EE	2
SE (*) ElementContent	3.0
CH ElementContent	3.1

Ensuite, l'événement de fin d'élément, correspondant à la fin de l'élément « person » est codé, en utilisant la production :

EE	2

Il est à noter que toutes les productions de la grammaire à l'exception de cette dernière production comprennent la description d'un événement, le code associé et la grammaire suivante à utiliser. Cette grammaire suivante est celle utilisé pour continuer le codage après le codage de l'événement compris dans la production.

Cependant, dans le cas d'un événement décrivant un début d'élément, les grammaires spécifiques à cet élément sont utilisées pour coder le contenu de l'élément. La grammaire suivante indiquée dans la production comprenant l'événement de début d'élément sera utilisée pour le codage après la fin de cet élément.

Aussi, la production comprenant l'événement de fin d'élément ne contient pas de grammaire suivante : la grammaire à utiliser pour coder la suite du document est celle qui avait été indiquée par la grammaire de l'élément parent dans la production utilisée pour coder l'événement de début de cet élément.

Si, par la suite, dans le document, le codeur rencontre un autre élément « person » similaire, cet élément va être codé à partir de cette grammaire.

Ainsi le premier événement correspondant au contenu de l'élément « person » est l'événement de début de l'élément « firstname ». Cet élément est codé avec la production :

SE (firstname) ElementContent 1

La production :

SE (*) ElementContent 3.0

correspond aussi à cet événement, mais est moins précise (elle ne précise pas le nom de l'élément). C'est donc la première production qui est utilisée.

L'encodeur code donc la priorité de cette production, à savoir « 1 », qui est codée avec les deux bits « 01 ». Il n'y a pas besoin de coder le nom de l'élément, puisque celui-ci est précisé par la production.

L'encodeur code ensuite le contenu de l'élément « firstname ».

Comme une production spécifique à l'événement de début de l'élément « firstname » existe déjà dans la grammaire, il n'est pas nécessaire d'ajouter une nouvelle production à la grammaire.

L'encodeur code ensuite, de manière similaire, l'événement de début de l'élément « lastname », en codant uniquement la priorité « 0 » avec les deux bits « 00 ».

Ainsi, pour le codage du deuxième élément « person » similaire au premier, le code généré est plus compact, puisqu'il n'est plus nécessaire de coder le nom des éléments contenu dans « person », ni littéralement (en codant l'intégralité de la chaîne de caractères), ni même à l'aide d'un index.

Si un schéma (que ce soit un document XML Schema ou un autre type de description) est disponible pour décrire le contenu de « person », à savoir un élément « firstname » et un élément « lastname », il est possible de construire, dès le départ, la grammaire générée après la fin du codage du premier élément « person ».

Cependant, si le schéma précise en outre que les éléments « firstname » et « lastname » doivent être ordonnés à l'intérieur de l'élément « person », « firstname » précédant « lastname », il est possible de générer les grammaires suivantes pour décrire le contenu de « person ».

ElementContent1 :	
SE (firstname) ElementContent2	0
EE	1
SE (*) ElementContent1	2.0
CH ElementContent1	2.1
ElementContent2 :	
SE (lastname) ElementContent3	0
EE	1
SE (*) ElementContent2	2.0
CH ElementContent2	2.1
ElementContent3 :	
EE	0
SE (*) ElementContent3	1.0
CH ElementContent3	1.1

Ces grammaires se servent de l'ordre connu d'apparition des éléments « firstname » et « lastname » pour séparer les productions en plusieurs grammaires et diminuer le nombre de productions par grammaire et, par conséquent, le nombre de bits nécessaires pour coder une priorité.

Ces grammaires peuvent même être réduites si l'on n'accepte pas les déviations par rapport au schéma. Ces grammaires deviennent alors :

ElementContent1 :	
SE (firstname) ElementContent2	0
ElementContent2 :	
SE (lastname) ElementContent3	0
ElementContent3 :	
EE	0

Dans ce cas, chaque grammaire ne contenant qu'une seule priorité, cette priorité n'a pas besoin d'être codée. Ainsi l'ensemble des événements décrivant le contenu de l'élément « person » tel que décrit ci-dessus est codé en 0 bits.

Une méthode pour améliorer le codage d'un document XML consiste à détecter et à utiliser des motifs (en anglais, « patterns ») dans le document

XML. Un motif représente un ensemble d'informations structurelles et, éventuellement, des informations de contenu du document XML. Le but de cette méthode est de coder les motifs répétés dans le document XML pour éviter de coder plusieurs fois les mêmes informations.

Ce type de méthode de codage est très efficace pour les documents XML contenant de nombreuses répétitions de structures identiques ou très similaires.

Un des buts de la présente invention est de proposer un système de codage similaire à Efficient XML capable d'utiliser des motifs pour améliorer la compacité des documents générés.

Les inventeurs ont, en effet, découvert que, comme l'ajout des motifs à un format XML Binaire permet une réduction importante de la taille des fichiers générés, il serait intéressant de pouvoir combiner un codage à base de motifs avec le format Efficient XML. Cependant, le format Efficient XML différant des autres formats par son utilisation de grammaires dynamiques, de nombreux problèmes se posent pour intégrer l'utilisation des motifs à ces grammaires dynamiques.

Selon un premier aspect, la présente invention vise un procédé de codage de données hiérarchisées organisées en une pluralité d'items pouvant être décrits par un ensemble d'événements en utilisant au moins une grammaire comportant des productions, ledit procédé comportant une étape d'adjonction d'au moins une production à une dite grammaire, caractérisé en ce que ladite étape d'adjonction comporte :

- une étape d'obtention d'un motif structurel représentant une pluralité d'informations au moins structurelles représentatives d'au moins un item à coder ;
- une étape d'extraction d'une pluralité d'événements décrivant ledit motif structurel obtenu ;
- une étape de création d'au moins une production comportant, ensemble, la pluralité d'événements extraits ; et
- une étape d'insertion de chaque dite production créée dans une dite grammaire.

Ainsi, on modifie la ou les grammaires pour améliorer l'encodage du contenu du document.

Selon des caractéristiques particulières, au cours de l'étape d'obtention d'un motif structurel, ledit motif structurel représente, en outre, des informations de contenu d'au moins un item à coder.

Selon des caractéristiques particulières, le procédé tel que succinctement exposé ci-dessus comporte, en outre, une étape d'association d'un code à chaque production créée.

Selon des caractéristiques particulières, au cours de l'étape d'association d'un code, on affecte à la nouvelle production un code à faible coût de codage.

Par exemple on affecte le code, ou la priorité, « 0 » à la nouvelle production.

Selon des caractéristiques particulières, le procédé tel que succinctement exposé ci-dessus comporte, en outre, une étape de codage d'au moins un motif structurel obtenu, en mettant en œuvre au moins une dite grammaire.

Selon des caractéristiques particulières, au cours de l'étape de codage d'au moins un motif structurel, au moins un motif structurel est codé en codant :

- un événement de début de description de motif,
- la pluralité des événements extrait décrivant ledit motif, et
- un événement de fin de description de motif.

Selon des caractéristiques particulières, au cours de l'étape de codage d'au moins un motif structurel obtenu, pour au moins un début d'élément contenu dans un motif structurel, on code une information indiquant si le contenu de cet élément est décrit dans le motif.

Selon des caractéristiques particulières, le procédé tel que succinctement exposé ci-dessus comporte, en outre, une étape de codage d'au moins un item à coder à l'aide d'un motif structurel le représentant.

Selon des caractéristiques particulières, au cours de l'étape de création d'au moins une production, on crée une production comportant au moins deux événements extraits décrivant le motif structurel obtenu.

Selon des caractéristiques particulières, au cours de l'étape de création d'au moins une production, on crée une production comportant un événement décrivant un premier item et au moins un autre événement décrivant un deuxième item contenu dans ledit premier item.

Ainsi, un motif peut être utilisé pour décrire un contenu non précisé au sein d'un autre motif.

Selon des caractéristiques particulières, le procédé tel que succinctement exposé ci-dessus comporte, pour des items rassemblés en une seule production, une étape de codage conjoint des dits items rassemblés en une seule production.

Selon des caractéristiques particulières, au cours de l'étape de création d'au moins une production, on crée une pluralité de productions comportant, chacune, au moins un événement de la pluralité d'événements obtenus décrivant le motif obtenu.

Selon des caractéristiques particulières, au cours de l'étape de création de la pluralité de productions, une première production de la pluralité de productions est insérée dans cette nouvelle grammaire et cette nouvelle grammaire est référencée par une deuxième production de la pluralité de productions.

Selon des caractéristiques particulières, le procédé tel que succinctement exposé ci-dessus comporte, préalablement à l'étape d'insertion d'au moins une production créée dans une grammaire, une étape de vérification de la présence de ladite production dans une dite grammaire, l'étape d'insertion n'étant réalisée que si ladite production n'est pas présente dans une dite grammaire.

Selon des caractéristiques particulières, le procédé tel que succinctement exposé ci-dessus comporte, en outre, une étape d'obtention d'au moins une grammaire initiale.

Selon des caractéristiques particulières, les données hiérarchisées sont exprimées sous la forme d'un document XML.

Selon un deuxième aspect, la présente invention vise un procédé de décodage de données hiérarchisées organisées en une pluralité d'items pouvant êtres décrits par un ensemble d'événements en utilisant au moins une grammaire comportant des productions, ledit procédé comportant une étape d'adjonction d'au moins une production à une dite grammaire, caractérisé en ce que ladite étape d'adjonction comporte :

- une étape d'obtention d'un motif structurel représentant une pluralité d'informations au moins structurelles représentatives d'au moins un item à décoder ;
- une étape d'extraction d'une pluralité d'événements décrivant ledit motif structurel obtenu ;
- une étape de création d'au moins une production comportant, ensemble, la pluralité d'événements extraits ; et
- une étape d'insertion de chaque dite production créée dans une dite grammaire.

Selon des caractéristiques particulières, le procédé de décodage tel que succinctement exposé ci-dessus comporte, en outre, une étape d'association d'un code à chaque production créée.

Selon un deuxième aspect, la présente invention vise un le procédé de décodage tel que succinctement exposé ci-dessus comporte, en outre, une étape de décodage d'au moins un motif structurel obtenu, en mettant en œuvre au moins une dite grammaire.

Selon un troisième aspect, la présente invention vise un dispositif de codage de données hiérarchisées organisées en une pluralité d'items pouvant êtres décrits par un ensemble d'événements en utilisant au moins une grammaire comportant des productions, ledit dispositif comportant un moyen d'adjonction d'au moins une production à une dite grammaire, caractérisé en ce que ledit moyen d'adjonction comporte :

- un moyen d'obtention d'un motif structurel représentant une pluralité d'informations au moins structurelles représentatives d'au moins un item à coder ;
- un moyen d'extraction d'une pluralité d'événements décrivant ledit motif structurel obtenu ;
- un moyen de création d'au moins une production comportant, ensemble, la pluralité d'événements extraits ; et
- un moyen d'insertion de chaque dite production créée dans une dite grammaire.

Selon un quatrième aspect, la présente invention vise un dispositif de décodage de données hiérarchisées organisées en une pluralité d'items pouvant êtres décrits par un ensemble d'événements en utilisant au moins une grammaire comportant des productions, ledit dispositif comportant un moyen d'adjonction d'au moins une production à une dite grammaire, caractérisé en ce que ledit moyen d'adjonction comporte, en outre :

- un moyen d'obtention d'un motif structurel représentant une pluralité d'informations au moins structurelles représentatives d'au moins un item à décoder ;
- un moyen d'extraction d'une pluralité d'événements décrivant ledit motif structurel obtenu ;
- un moyen de création d'au moins une production comportant, ensemble, la pluralité d'événements extraits ; et
- un moyen d'insertion de chaque dite production créée dans une dite grammaire.

Selon un cinquième aspect, la présente invention vise un programme d'ordinateur chargeable dans un système informatique, ledit programme contenant des instructions permettant la mise en œuvre du procédé de codage tel que succinctement exposé ci-dessus et/ou du procédé de décodage tel que succinctement exposé ci-dessus.

Selon un sixième aspect, la présente invention vise un support d'informations lisibles par un ordinateur ou un microprocesseur, amovible ou non, conservant des instructions d'un programme informatique, caractérisé en

ce qu'il permet la mise en œuvre du procédé de codage tel que succinctement exposé ci-dessus et/ou du procédé de décodage tel que succinctement exposé ci-dessus.

Il est à noter que l'ajout des motifs au format Efficient XML est intéressant quelle que soit la manière dont les grammaires initiales sont créées. Si les grammaires initiales sont les grammaires génériques (décrites dans la spécification et créées à partir de la syntaxe XML), les motifs permettent de décrire plus précisément le contenu d'un élément. Si les grammaires initiales ont été créées à partir d'un Schéma XML (ou d'une autre source d'information similaire) décrivant la structure générale du document XML, les grammaires décrivent déjà le contenu de chaque élément XML. Cependant, un Schéma XML devant décrire l'ensemble des possibilités pouvant survenir dans un document XML, les grammaires peuvent comporter un nombre plus ou moins important d'items XML optionnels. L'ajout des motifs permet de préciser les options utilisées par un document XML donné, et donc d'adapter au document XML des grammaires génériques correspondant à sa classe de documents.

Les avantages, buts et caractéristiques procédé de décodage, de ces dispositifs, de ce programme et de ce support d'information étant similaires à ceux du procédé de codage objet de la présente invention, tel que succinctement exposé ci-dessus, ils ne sont pas rappelés ici.

D'autres avantages, buts et caractéristiques de la présente invention ressortiront de la description qui va suivre faite, dans un but explicatif et nullement limitatif en regard des dessins annexés, dans lesquels :

- la figure 1 représente, schématiquement, un mode de réalisation particulier du dispositif de codage et du dispositif de décodage objets de la présente invention,

- la figure 2 représente, sous forme d'un logigramme, un mode de réalisation particulier du procédé de codage objet de la présente invention,

- la figure 3 représente, sous forme d'un logigramme, des étapes mises en œuvre dans un mode de réalisation particulier du procédé de décodage objet de la présente invention,

- la figure 4 représente, sous forme d'un logigramme, des étapes de mises en œuvre dans un premier mode de réalisation du procédé d'enrichissement de grammaires en fonction d'un motif structurel et

- la figure 5 représente, sous forme d'un logigramme, des étapes de mises en œuvre dans un deuxième mode de réalisation du procédé d'enrichissement de grammaires en fonction d'un motif structurel.

Avant de décrire la présente invention en regard des figures, on donne, ci-après, une définition détaillée des motifs.

Un motif structurel est un ensemble d'informations structurelles contenues dans un document XML. Ces informations structurelles peuvent comprendre le type d'un item XML (élément, attribut, texte, commentaire...), la relation entre deux items XML (parent – enfant, grand-parent – petit-enfant, frères...), l'ordre des sous-items composant un élément... En outre, un motif structurel peut être étendu pour contenir des informations de contenu comme la valeur d'un attribut, la valeur d'un contenu textuel, la valeur d'un commentaire...

De manière générale, la méthode pour construire un motif structurel consiste à extraire des informations structurelles d'un document XML, de regrouper les informations structurelles décrivant une même partie du document (par exemple un élément), puis de détecter quels sont les ensembles d'informations structurelles se retrouvant régulièrement dans le document et quelles sont les valeurs de contenu fréquemment associées à ces ensembles d'informations structurelles. Ce sont ces ensembles d'informations structurelles répétés dans le document, avec leurs valeurs de contenu fréquentes qui forment les motifs structurels utilisés dans le codage du document. L'intérêt de l'utilisation d'un motif structurel est de ne décrire qu'une seule fois la structure correspondante dans le codage du fichier.

En variante, deux ensembles d'informations structurelles ne possédant que peu de différences peuvent être combinés en un seul motif structurel. L'avantage d'une telle combinaison est de ne décrire qu'un seul motif structurel.

Le document XML suivant va servir d'exemple pour la construction de motifs structurels.

```
<list>
<person>
        <firstname>Jean</firstname>
        <lastname>Dupont</lastname>
</person>
<person>
        <firstname>Marie</firstname>
        <lastname>Durant</lastname>
</person>
<person>
        <firstname>Pierre</firstname>
        <lastname>Martin</lastname>
</person>
<person>
        <firstname>Yann</firstname>
        <lastname>Kermarec</lastname>
        <city>Rennes</city>
</person>
<person>
        <firstname>Brieuc</firstname>
        <lastname>Gwendu</lastname>
        <city>Rennes</city>
</person>
</list>
```

On remarque que l'élément « person » a deux structures proches au sein de ce document. La première consiste en deux sous-éléments, tandis que la deuxième consiste en trois sous-éléments. Il est donc possible de construire deux motifs représentant ces structures. Le premier motif, « motif1 » est :

```
person
  motif
  motif
```

Ce motif décrit un élément « person » contenant deux sous-éléments, sans préciser quels sont ces sous-éléments : ces deux sous-éléments sont décrits dans « motif1 » par l'indication « motif » qui précise que l'item contenu est un sous-élément dont la structure n'est pas précisée.

Le deuxième motif, « motif2 », est :

```
person
   motif
   motif
   motif
```

Ce motif décrit un élément « person » contenant trois sous-éléments, sans préciser quels sont ces sous-éléments.

Dans les deux cas, les sous-éléments contenus dans l'élément « person » sont toujours les mêmes. Une variante du motif1 peut donc être construite pour décrire plus précisément l'élément « person ». Ce motif, « motif3 », est :

```
person
   firstname
         *
   lastname
         *
```

Ce motif décrit un élément « person » contenant deux sous-éléments, « firstname » et « lastname », sans préciser le contenu de ces sous-éléments, ce qui est représenté par la valeur « * » comme contenu de ces sous-éléments.

Comme le contenu de ces sous-éléments est toujours une valeur textuelle, il est possible de construire un autre motif, « motif4 », le précisant :

```
person
   firstname
         text
   lastname
         text
```

Ce motif décrit un élément « person » contenant deux sous-éléments, « firstname » et « lastname », ces deux sous-éléments ayant un contenu textuel.

Il est bien sûr possible de construire des motifs similaires à partir du motif2. Ainsi, un motif décrivant mieux la structure des deux derniers éléments « person », « motif5 », est :

person
  firstname
    text
  lastname
    text
  city
    Rennes

Ce motif décrit un élément « person » contenant trois sous-éléments, « firstname », « lastname » et « city », « firstname » et « lastname » ayant un contenu textuel, et « city » ayant un contenu textuel constant : « Rennes ».

Enfin, les motifs « motif4 » et « motif5 » peuvent être combinés en un motif « motif6 » :

person
  firstname
    text
  lastname
    text
  city?
    Rennes

Ce motif « motif6 »décrit un élément « person » contenant au moins deux sous-éléments, « firstname » et « lastname » qui contiennent une valeur textuelle, et un troisième sous-élément optionnel, « city », ayant un contenu textuel constant : « Rennes ».

Dans le cas d'un document plus complexe, où, par exemple, un élément « address » supplémentaire ayant un contenu très variable (avec des sous-éléments optionnels...) est présent dans l'élément « person », les motifs

peuvent être utilisés récursivement. Ainsi, un premier motif va décrire l'élément « person », en précisant qu'il contient un sous-élément « address » sans définir ce sous-élément. Un ou plusieurs autres motifs pourront préciser les différentes structures que peut prendre ce sous-élément « address ». Ainsi, un motif peut être utilisé pour décrire un contenu non précisé au sein d'un autre motif.

On donne, ci-dessous, un premier exemple de grammaires et un premier mode de réalisation du procédé de codage objet de la présente invention. Dans ce premier mode de réalisation, une modification des grammaires pour représenter des motifs consiste à ajouter une production contenant une séquence d'événements décrivant le motif. Cette méthode de modification est décrite précisément en référence à la figure 4.

La grammaire initiale servant à décrire le contenu de l'élément « person » est :

ElementContent :

EE	0
SE (*) ElementContent	1.0
CH ElementContent	1.1

Il s'agit de la grammaire générique décrivant le contenu d'un élément XML (par soucis de simplification, certaines productions correspondant à des événements peu fréquents ont été supprimés, et l'exemple ne tient pas compte de la grammaire StartTagContent utilisée pour décrire le contenu de la balise de début d'un élément).

Le motif « motif1 » peut être intégré dans cette grammaire en ajoutant une production :

SE (*) SE (*) EE	n

qui décrit le début d'un premier sous-élément non défini, le début d'un deuxième sous-élément non défini et la fin de l'élément « person ». Lors de l'utilisation de cette production, la priorité de cette production doit être codée, suivie du nom du premier sous-élément. Ce premier sous-élément est alors codé à l'aide des grammaires qui lui sont associées. Puis le nom du deuxième sous-élément est codé. Puis, le deuxième sous-élément est codé à l'aide des

grammaires qui lui sont associées. Enfin, pour l'événement de fin d'élément « person », rien n'est à coder.

Il est à noter que, lors du codage de l'un de ces sous-éléments, s'il existe au sein des grammaires correspondant à ce sous-élément un motif associé au sous-élément à coder, ce motif peut être utilisé pour coder le sous-élément.

Le motif « motif3 » est intégré dans la grammaire avec une production légèrement différente :

SE (firstname) SE (lastname) EE n

Lors de l'utilisation de cette production, il n'est pas utile de coder le nom des deux sous-éléments, puisque ceux-ci sont définis dans la production.

Le motif « motif4 » est intégré dans la grammaire avec la production :

SEI (firstname) CH EE SEI (lastname) CH EE EE n

L'utilisation de cette production diffère par certains points des précédentes. En premier lieu, sa priorité est codée. Ensuite, le nom du premier sous-élément n'a pas à être codé puisqu'il est défini dans la production. Pour coder le sous-élément, puisque l'événement décrivant le début du sous-élément est SEI, et non SE, la même production continue d'être utilisée, et non l'ensemble des grammaires correspondant à l'élément « firstname ». Ainsi, le contenu textuel de « firstname » est codé. Puis « EE » signifie que le sous-élément « firstname » est terminé. Le sous-élément « lastname » est codé de la même manière. Le dernier « EE » signifie que l'élément « person » est terminé.

Le motif « motif5 » peut être intégré de manière similaire au « motif4 » :

SEI (firstname) CH EE
SEI (lastname) CH EE
SEI (city) CHC (Rennes) EE EE n

L'utilisation de cette production est similaire à celle correspondant à « motif4 ». La seule différence est que le contenu textuel de « city » est représenté par « CHC (Rennes) », ce qui signifie que le contenu textuel de « city » est constant et a la valeur « Rennes ». Lors du codage du sous-élément « city », il n'est donc pas utile de coder son contenu textuel.

En revanche, le motif « motif6 » ne peut être intégré dans une production : il comporte une possibilité de variation, ce qui n'est pas possible au sein d'une production. Néanmoins, si le motif « motif6 » ne peut être représenté, ses deux variantes (« motif4 » et « motif5 ») peuvent chacune être représentées par une production.

Cette première méthode de modification des grammaires pour représenter des motifs permet donc de représenter les motifs ne comportant pas d'options comme une production.

Il est à noter que, dans les exemples, le décodage s'effectue de manière symétrique par rapport au codage.

Il est aussi à noter que cette méthode peut aussi s'appliquer à une grammaire construite à partir d'un Schéma XML décrivant la structure du document.

On donne, ci-dessous, un deuxième exemple de modification des grammaires pour représenter des motifs consiste à ajouter un ensemble de grammaire, chacune des nouvelles grammaires décrivant un événement du motif. Cette méthode est décrite en référence à la figure 5.

La grammaire initiale servant à décrire le contenu de l'élément « person » est la même que précédemment :

ElementContent :

EE	0
SE (*) ElementContent	1.0
CH ElementContent	1.1

Le motif « motif1 » peut être intégré dans cette grammaire en ajoutant une production à cette grammaire initiale :

SE (*) Motif1-1	n

Cette production décrit le début d'un premier sous-élément non précisé, et précise que la grammaire à utiliser pour coder la suite du contenu de l'élément « person » est « Motif1-1 ».

Outre la production précédemment décrite, plusieurs grammaires sont ajoutées pour décrire la suite du motif :

Motif1-1 :

SE (*) Motif1-2 0

Motif1-2 :

EE 0

La grammaire « Motif1-1 » décrit un sous-élément non précisé (qui correspond au deuxième sous-élément contenu dans le motif) et indique que la grammaire à utiliser pour coder la suite du contenu de l'élément « person » est « Motif1-2 ». La grammaire « Motif1-2 » décrit la fin de l'élément « person ».

Lors de l'utilisation de la nouvelle production et des nouvelles grammaires, pour coder une instance de « motif1 », l'encodeur effectue un ensemble d'opérations. Il commence par coder la priorité de la nouvelle production, suivie du nom du premier sous-élément. Ce premier sous-élément est alors codé à l'aide des grammaires qui lui sont associées. Ensuite, l'encodeur utilise la grammaire « Motif1-1 » pour coder le deuxième sous-élément. Comme cette grammaire ne possède qu'une seule production, il n'est pas nécessaire de coder sa priorité. En revanche, le nom de ce deuxième sous-élément est codé. Le deuxième sous-élément est codé à l'aide des grammaires qui lui sont associées. Enfin, l'encodeur utilise la grammaire « Motif1-2 » pour coder la fin de l'élément « person ». Comme cette grammaire ne possède qu'une seule production, là encore, il n'est pas nécessaire de coder sa priorité.

De manière similaire, « motif3 » est intégré aux grammaires en ajoutant la production suivante dans la grammaire « ElementContent » de l'élément « person » :

SE (firstname) Motif3-1 n

Et en créant les grammaires suivantes :

Motif3-1 :

SE (lastname) Motif3-2 0

Motif3-2 :

EE 0

Lors du codage, ces grammaires sont utilisées comme décrit précédemment.

Pour le motif « motif4 », la production à ajouter est la suivante :

SEI (firstname) Motif4-1 n

Et les grammaires à créer sont les suivantes :

Motif4-1 :

CH Motif4-2 0

Motif4-2 :

EE Motif4-3 0

Motif4-3 :

SEI (lastname) Motif4-4 0

Motif4-4 :

CH Motif4-5 0

Motif4-5 :

EE Motif4-6 0

Motif4-6 :

EE 0

Comme dans le cas de la première méthode, les événements SEI signifient que la même production continue d'être utilisée pour coder le contenu de l'élément correspondant. Ainsi dans le cas de la production ajoutée à la grammaire « ElementContent », le contenu du sous-élément « firstname » est codé à l'aide de la grammaire « Motif4-1 » et des suivantes.

Le motif « motif5 » est intégré en utilisant la production :

SEI (firstname) Motif5-1 n

Les grammaires à créer sont :

Motif5-1 :

CH Motif5-2 0

Motif5-2 :

EE Motif5-3 0

Motif5-3 :

SEI (lastname) Motif5-4 0

Motif5-4 :

CH Motif5-5 0

Motif5-5 :

EE Motif5-6 0

Motif5-6 :

SEI (city) Motif5-7 0

Motif5-7 :

CHC (Rennes) Motif5-8 0

Motif5-8 :

EE Motif5-9 0

Motif5-9 :

EE 0

Comme précédemment, l'événement « CHC (Rennes) » signifie que le contenu textuel a une valeur constante « Rennes ».

Le motif « motif6 » peut être intégré en combinant les règles correspondant aux motifs « motif4 » et « motif5 ». La production ajoutée est :

SEI (firstname) Motif6-1 n

Les grammaires à créer sont :

Motif6-1 :

CH Motif6-2 0

Motif6-2 :

EE Motif6-3 0

Motif6-3 :

SEI (lastname) Motif6-4 0

Motif6-4 :

CH Motif6-5 0

Motif6-5 :

EE Motif6-6 0

Motif6-6 :

SEI (city) Motif6-7 0

EE 1

Motif6-7 :

CHC (Rennes) Motif6-8 0

Motif6-8 :

EE Motif6-9 0

Motif6-9 :

EE 0

Il est à noter que, dans ce cas, la grammaire « Motif6-6 » comporte deux productions correspondant à chacune des options possibles pour le motif « motif6 » : présence du sous-élément « city » ou non.

De manière similaire, cette deuxième méthode permet de coder aisément un motif comportant un choix de valeur pour un contenu donné. Ainsi, si, dans le motif « motif6 », le contenu du sous-élément « city » pouvait prendre les valeurs « Rennes », « Nantes » et « Brest », la grammaire « Motif6-7 » deviendrait :

« Motif6-7 » :

CHC (Rennes) Motif6-8	0
CHC (Nantes) Motif6-8	1
CHC (Brest) Motif6-8	2

Toujours de manière similaire, cette deuxième méthode permet aussi de coder aisément des valeurs fréquentes pour un contenu donnée. Ainsi, si dans le motif « motif6 », le contenu du sous-élément « city » prend souvent les valeurs « Rennes » et « Nantes », mais peut aussi prendre d'autres valeurs moins fréquentes, la grammaire « Motif6-7 » devient :

« Motif6-7 » :

CHC (Rennes) Motif6-8	0
CHC (Nantes) Motif6-8	1
CH Motif6-8	2

Il est à noter que, comme pour la première méthode, cette méthode peut aussi s'appliquer à un ensemble de grammaires construites à partir d'un Schéma XML décrivant la structure du document.

Ci-dessous, on décrit l'ensemble des événements utilisés par Efficient XML pour décrire et coder un document XML.

Notation	Description	Contenu
SD	Start Document	
ED	End Document	
SE (qname)	Start Element	QName
EE	End Element	
AT (qname)	Attribute	QName, Value

CH	Characters	Value
NS	Namespace Declaration	Prefix, URI
CM	Comment	Value
PI	Processing instruction	Name, Value
DT	DOCTYPE	Name, Public, System
EDT	End DOCTYPE	
ER	Entity Reference	Name

Cependant, afin de pouvoir ajouter l'utilisation des motifs à Efficient XML, dans un mode de réalisation, la mise en œuvre de la présente invention ajoute quelques autres événements spécifiques utiles à l'utilisation des motifs. Ces nouveaux événements sont décrits ci-dessous.

Notation	Description	Contenu
SEI (qname)	Start Element Inlined	QName
ATC (qname, value)	Constant Attribute	QName, Value
CHC (value)	Constant Characters	Value
SP	Start Pattern	
EP	End Pattern	
OPT	Option	Number events
OR	Choice	Number

Les événements utilisés par Efficient XML correspondent pour la plupart aux événements utilisés de façon classique avec XML, en particulier par les parseurs dit « streaming » qui génèrent, à partir d'un document XML, une suite d'événements le décrivant. Pour chaque événement, les listes d'événements données ci-dessus décrivent la notation utilisée pour cet événement, sa description et les valeurs contenues dans cet événement.

« SD (Start Document) » correspond au début du document XML.

« ED (End Document) » correspond à la fin du document XML.

« SE (qname) (Start Element) » correspond au début d'un élément XML nommé « qname ». L'événement « SE (qname) » peut prendre deux formes. Dans la première forme, « SE (qname) », le nom de l'élément XML, qname, est précisé dans l'événement. Dans la deuxième forme, « SE (*) », le

nom de l'élément XML n'est pas précisé dans l'événement. Il doit donc être codé à chaque occurrence de l'événement.

« EE (End Element) » correspond à la fin d'un élément XML : il s'agit de la fin du dernier élément XML débuté.

« AT (qname) (Attribute) » correspond à un attribut d'un élément XML nommé « qname ». L'événement « AT (qname) » peut prendre deux formes. Dans la première forme, « AT (qname) », le nom de l'attribut XML, qname, est précisé dans l'événement. Dans la deuxième forme, « AT (*) », le nom de l'attribut XML n'est pas précisé dans l'événement. Il doit donc être codé à chaque occurrence de l'événement. En outre, pour les deux formes, pour chaque occurrence de l'événement, la valeur (« Value ») de l'attribut doit être codée.

« CH (Characters) » correspond à un contenu textuel. Pour chaque occurrence de cet événement, le contenu textuel (« Value ») doit être codé.

« NS (Namespace declaration) » correspond à une déclaration d'espace de nommage. Pour chaque occurrence de cet événement, l'URI de l'espace de nommage et le préfixe (« Prefix ») qui lui est associé doivent être codés.

« CM (Comment) » correspond à un commentaire. Pour chaque occurrence de cet événement, le contenu du commentaire (« Value ») doit être codé.

« PI (Processing Instruction) » correspond à une instruction de traitement. Pour chaque occurrence de cet événement, le nom (« Name ») et la valeur (« Value ») de l'instruction de traitement doivent être codés.

« DT (DOCTYPE) » correspond à un début de type de document. Pour chaque occurrence de cet événement, le nom (« Name »), et les identifiants « public » (« Public ») et « système » (« System ») du type de document doivent être codés.

« EDT (End DOCTYPE) » correspond à une fin de type de document.

« ER (Entity Reference) » correspond à une référence d'entité. Pour chaque occurrence de cet événement, le nom de l'entité (« Name ») doit être codé.

La mise en œuvre de la présente invention ajoute quelques autres événements.

« SEI (qname) (Start Element Inlined) » correspond à un début d'un élément XML nommé « qname ». Comme pour « SE (qname) », l'événement peut préciser ou non le nom de l'élément. Par contre, dans le cas de l'élément « SE (qname) », la grammaire à utiliser pour coder le contenu de l'élément « qname » est celle correspondant à cet élément. Dans le cas de l'élément « SEI (qname) », la grammaire à utiliser pour coder le contenu de l'élément « qname » est indiquée dans la production contenant cet événement. Cet événement permet à une grammaire de décrire non seulement le contenu d'un élément, mais aussi le contenu d'un sous-élément de cet élément.

« ATC (qname, value) (Constant Attribute) » correspond à un attribut XML nommé « qname » ayant une valeur constant « value ». Cet événement permet de définir au sein même de l'événement son nom et sa valeur et permet, dans le cas d'attributs avec des valeurs souvent répétées, de ne pas nécessiter le codage de la valeur de l'attribut en plus du codage de l'événement lui-même.

« CHC (value) (Constant Character) » correspond à un contenu textuel constant, ayant pour valeur « value ». Le contenu textuel est défini au sein même de l'événement et n'a donc pas à être codé pour chaque occurrence de l'événement. Cet événement est utile en cas de contenu textuel souvent répété.

« SP (Start Pattern) » correspond au début de la description d'un motif.

« EP (End Pattern) » correspond à la fin de la description d'un motif.

« OPT (Option) » correspond à une partie optionnelle d'un motif. Après cet événement, le nombre d'événements optionnels dans le motif (« Number events ») est codé.

« OR (Choice) » correspond à un choix entre plusieurs possibilités pour l'événement suivant. Après cet événement, le nombre de choix (« Number ») est codé, puis chacun des choix est codé de manière séquentielle. Il est à noter que cet événement permet à la fois de coder un choix entre plusieurs valeurs différentes, mais aussi un choix entre plusieurs valeurs fréquentes prédéfinies et une valeur non précisée représentant les autres cas.

Les événements « SP », « EP », « OPT » et « OR » sont préférentiellement utilisés si la définition des motifs est codée avec l'ensemble du document XML. Pour pouvoir coder les motifs, la production suivante doit être ajoutée aux grammaires de description du contenu d'un élément :

SP	n.m

De préférence, cette production est ajoutée avec une priorité faible, c'est-à-dire comportant plusieurs niveaux.

Un motif est codé par exemple avec la grammaire suivante :

PatternContent:

EP	0
SE (*) PatternContent	1.0
SEI (*) PatternContent	1.1
EE PatternContent	1.2
AT (*) PatternContent	1.3
ATC (*, *) PatternContent	1.4
CH PatternContent	1.5
CHC (*) PatternContent	1.6
OPT PatternContent	1.7
OR PatternContent	1.8

Il est possible de diviser cette grammaire en plusieurs grammaires pour optimiser le codage du motif.

On observe, en figure 1, un mode particulier de réalisation du dispositif de codage et du dispositif de décodage objets de la présente invention, 100 et différents périphériques adaptés à implémenter chaque aspect de la présente invention. Dans le mode de réalisation illustré en figure 1, le

dispositif 100 est un micro-ordinateur de type connu connecté, dans le cas du codeur, par le biais d'une carte graphique 104, à un moyen d'acquisition ou de stockage de documents structurés 101, par exemple une interface avec un réseau informatique local ou avec une mémoire, adapté à fournir des données d'un document structuré.

Le dispositif 100 comporte une interface de communication 118 reliée à un réseau 134 apte à transmettre, en entrée, des données numériques à coder ou à décoder et, en sortie, des données codées ou décodées par le dispositif 100. Le dispositif 100 comporte également un moyen de stockage 112, par exemple un disque dur, et un lecteur 114 de disquette 116. La disquette 116 et le moyen de stockage 112 peuvent contenir des données à coder ou à décoder, des données codées ou décodées et un programme informatique adapté à implémenter le procédé de codage ou le procédé de décodage objets de la présente invention.

Selon une variante, le programme permettant au dispositif de mettre en œuvre la présente invention est stocké en mémoire morte ROM (acronyme de « read only memory » pour mémoire non réinscriptible) 106. Selon une autre variante, le programme est reçu par l'intermédiaire du réseau de communication 134 avant d'être stocké.

Le dispositif 100 possède un écran 105 permettant de visualiser les données à coder ou décodées ou servant d'interface avec l'utilisateur pour paramétrer certains modes d'exécution du dispositif 100, à l'aide d'un clavier 110 et/ou d'une souris par exemple.

Une unité centrale CPU (acronyme de « central processing unit ») 103 exécute les instructions du programme informatique et de programmes nécessaires à son fonctionnement, par exemple un système d'exploitation. Lors de la mise sous tension du dispositif 100, les programmes stockés dans une mémoire non volatile, par exemple la mémoire morte 106, le disque dur 112 ou la disquette 116, sont transférés dans une mémoire vive RAM (acronyme de « random access memory » pour mémoire à accès aléatoire) 108 qui contiendra alors le code exécutable du programme objet de la présente invention ainsi que des registres pour mémoriser les variables nécessaires à sa mise en œuvre.

Bien entendu, la disquette 116 peut être remplacée par tout support d'information amovible, tel que disque compact, clé ou carte mémoire. De manière plus générale, un moyen de stockage d'information, lisible par un ordinateur ou par un microprocesseur, intégré ou non au dispositif, éventuellement amovible, mémorise un programme objet de la présente invention. Un bus de communication 102 permet la communication entre les différents éléments inclus dans le dispositif 100 ou reliés à lui. La représentation, en figure 1, du bus 102 n'est pas limitative et notamment l'unité centrale 103 est susceptible de communiquer des instructions à tout élément du dispositif 100 directement ou par l'intermédiaire d'un autre élément du dispositif 100.

Le dispositif décrit ici et, particulièrement, l'unité centrale 103, sont susceptibles d'implémenter tout ou partie des traitements décrits en regard des figures 2 à 5, pour mettre en œuvre chaque procédé objet de la présente invention et constituer chaque dispositif objet de la présente invention.

La figure 2 décrit, sous forme d'un logigramme, des étapes d'un mode de réalisation particulier du procédé de codage objet de la présente invention. Ces étapes décrivent le traitement appliqué à un événement XML, obtenu, par exemple, à partir d'un parseur SAX (acronyme de « Simple API for XML », pour Interface de programmation simple pour XML) ou d'un parseur StAX (acronyme de « Streaming API for XML », pour Interface de programmation pour XML sous forme de flux) pour encoder cet événement.

La première étape, 200, consiste à obtenir l'événement avec l'ensemble des informations qui le composent.

A partir de cet événement (et des événements précédents et éventuellement d'événement suivants si l'encodeur peut regarder en avant dans le fichier), l'encodeur tente de détecter un motif, au cours d'une étape 210. La détection de motif peut se faire selon plusieurs types d'algorithmes. Un premier type d'algorithme détecte les motifs à partir d'une analyse d'une partie ou de l'ensemble du fichier, en regroupant les structures similaires. Un autre type d'algorithme détecte les motifs à partir d'une analyse simplifiée portant uniquement sur le début du fichier. Ce second type d'algorithme est

particulièrement adapté pour traiter des fichiers en mode streaming. L'ensemble de ces algorithmes a pour but de détecter les ensembles d'informations structurelles et de contenu se répétant fréquemment dans le document XML.

Une variante de l'invention consiste à effectuer l'étape 210 au préalable à l'encodage du fichier en utilisant un algorithme du premier type.

Puis, au cours d'une étape 220, on détermine si un nouveau motif est utile pour encoder l'événement obtenu précédemment. A cet effet, soit on détermine si un motif a été détecté en association à l'événement lors de l'étape 210 et si ce motif n'a pas encore été défini dans l'encodage du fichier, soit on détermine si le motif remplit des critères prédéfinis d'utilisation. Ces critères servent à déterminer si le motif permet de générer un document encodé plus compact. Un tel critère peut être, par exemple, le nombre d'instances du motif déjà rencontrées dans le document XML.

Si le résultat de l'étape 220 est positif, au cours d'une étape 230, on code la définition du motif. Une méthode pour coder la définition du motif consiste à coder un événement « SP » de début de description de motif, à coder l'ensemble des événements composant le motif, et à coder un événement « EP » de fin de description de motif. Les événements composant le motif sont codés comme s'ils décrivaient une instance du motif, avec quelques variations. Tout d'abord, dans le cadre du codage d'un motif, il n'est pas utile de coder le contenu variable des événements composant le motif. Aussi, lors du codage des événements, tout contenu variable est codé comme valeur nulle. Ensuite, un motif peut soit décrire le contenu d'un sous-élément, soit décrire uniquement la présence de ce sous-élément sans décrire son contenu. Aussi, après chaque événement de début d'élément un bit définit si cet élément est décrit ou non dans le motif.

Il est à noter qu'il est tout à fait possible d'utiliser, au sein de la description d'un motif, un événement correspondant à un motif déjà intégré dans une grammaire.

Une première variante consiste à ajouter de nouveaux événements spécifiques au codage des motifs, ces événements étant spécialisés pour indiquer si leur contenu est variable ou non. Ainsi, dans le cas d'un événement

d'un motif ayant un contenu variable, un événement spécialisé est utilisé, ce qui évite de devoir coder le contenu variable comme valeur nulle. Dans le cas d'un événement d'un motif ayant un contenu non variable, un autre événement spécialisé est utilisé, autorisant le codage du contenu constant.

Une deuxième variante de cette méthode consiste à coder conjointement le motif et sa première instance. Dans ce cas, l'ensemble des événements composant l'instance du motif est codé de manière classique, précédé d'un événement « SP » de début de description de motif et suivi d'un événement « EP » de fin de description de motif. En outre, pour indiquer quels sont les contenus constants et les contenus variables du motif, l'événement « EP » de fin de description de motif est suivi d'un ensemble de variables booléennes indiquant, pour chaque contenu, s'il est constant ou non.

Une autre variante consiste à permettre de coder un motif en référence à un autre motif déjà codé, en ne codant que les différences entre ces deux motifs. A cet effet, après l'événement « SP » de début de description de motif, la priorité de la production correspondant à cet autre motif est codée. Peuvent être codés ensuite les événements supprimés dans le nouveau motif, en codant pour chaque événement une valeur booléenne. Peuvent être codés aussi les contenus différents dans le nouveau motif, en codant pour chaque événement une valeur booléenne, l'ensemble de ces valeurs booléennes étant suivi par les valeurs de contenu différentes. Peuvent être codés aussi des événements ajoutés dans le nouveau motif, en codant le nombre d'événement ajoutés et pour chaque événement sa position, ainsi que l'événement lui-même en utilisant la grammaire correspondant à sa position au sein du motif.

Une variante à l'étape 230 est mise en œuvre dans le cas du second type d'algorithme de détection de motif. Si le motif détecté au cours de l'étape 210 l'a été uniquement à partir des événements précédant l'événement obtenu (sans inclure cet événement obtenu) et si les méthodes de détection de l'étape 210 et les critères d'utilisation du motif de l'étape 220 sont connus par l'encodeur et par le décodeur, alors il n'est pas nécessaire de coder le motif. Le motif détecté et utilisé par l'encodeur sera détecté et utilisé par le décodeur de manière identique. Il est possible de faire dépendre l'étape 210 et l'étape 220

de quelques paramètres variables, mais dans ce cas, ces paramètres devront être transmis entre l'encodeur et le décodeur au sein du fichier encodé.

Après l'étape 230, au cours d'une étape 240, on modifie les grammaires utilisées par l'encodeur comme décrit en référence aux figures 4 et 5.

Après l'étape 240 ou après l'étape 220 si son résultat est négatif, au cours d'une étape 250, on détermine si l'événement obtenu fait partie d'une séquence d'événements rassemblés au sein d'une même production.

Si le résultat de l'étape 250 est négatif, c'est-à-dire si l'événement ne fait pas partie d'une séquence d'événements d'une même production, on encode l'événement selon le format Efficient XML, au cours d'une étape 260, puis on modifie les grammaires comme spécifié par le format Efficient XML, au cours d'une étape 270. En particulier, cette étape change la grammaire courante selon les règles spécifiées par le format Efficient XML.

Si le résultat de l'étape 250 est positif, c'est-à-dire si l'événement fait partie d'une séquence d'événements d'une même production, au cours d'une étape 280, on encode l'événement comme appartenant à cette séquence. Deux cas peuvent se présenter pour la réalisation de l'étape 280. Si l'événement est le premier événement de cette séquence, on encode la priorité de cette séquence, puis on encode les éventuels contenus de l'événement non décrits dans la production. Si l'événement n'est pas le premier événement de cette séquence, on encode uniquement les éventuels contenus de l'événement non décrits dans la production.

Puis, au cours d'une étape 290, on procède à une modification des grammaires. En particulier, cette étape change la grammaire courante selon les règles spécifiées par le format Efficient XML. Cependant, dans le cas d'un événement SE (*) appartenant à une production contenant une séquence d'événements, aucune nouvelle production n'est créée, contrairement aux règles spécifiées par le format Efficient XML.

Dans tous les cas, l'encodage se poursuit en vérifiant s'il reste un événement à encoder, au cours d'une étape 295, et, si c'est le cas, en retournant à l'étape 200. Sinon, l'encodage est achevé.

La Figure 3 décrit, sous forme de logigramme, le décodage d'un document XML, dans un mode de réalisation du procédé de décodage objet de la présente invention. On décrit, en figure 3, le traitement appliqué à un événement encodé dans le fichier en cours de décodage.

Au cours d'une étape 300, on obtient une priorité à partir du fichier codé. Cette obtention s'effectue comme décrit dans la spécification Efficient XML en lisant la priorité en fonction de la grammaire courante. A partir de cette priorité, on obtient la production utilisée par l'encodeur.

Puis, au cours d'une étape 310, on détermine si la production correspond à la description d'un motif. Si oui, au cours d'une étape 320 on décode le motif et, au cours d'une étape 330, on modifie les grammaires, comme décrit en référence aux figures 4 et 5. Puis on passe à l'étape 395 décrite plus loin.

Si le résultat de l'étape 310 est négatif, au cours d'une étape optionnelle 340, on effectue une détection de motif. Cette étape 340 est présente dans le cas d'un algorithme de détection de motif de second type (voir plus haut les différents types d'algorithme de détection), si les motifs ne sont pas transmis entre le codeur et le décodeur. Cette étape 340 permet au décodeur de découvrir les motifs utilisés par le dispositif de codage en réalisant le même traitement que le dispositif de codage sur les événements déjà reçus.

Que l'étape 340 soit présente ou pas, on détermine ensuite si la production correspond à une séquence d'événements, au cours d'une étape 350. Si ce n'est pas le cas, on décode le reste de l'événement, c'est-à-dire les éventuels contenus non spécifiés par la production, au cours d'une étape 360, puis on modifie les grammaires comme spécifié par le format Efficient XML, au cours d'une étape 270. En particulier, cette étape 370 change la grammaire courante selon les règles spécifiées par le format Efficient XML.

Si le résultat de l'étape 350 est positif, on décode l'ensemble de la séquence, au cours d'une étape 380. A cet effet, pour chaque événement contenu dans la séquence, on décode les éventuels contenus non spécifiés dans la production. Puis, au cours d'une étape 390, on procède à une modification des grammaires. En particulier, cette étape 390 change la

grammaire courante selon les règles spécifiées par le format Efficient XML. Par contre, dans le cas d'un événement SE (*) appartenant à une production contenant une séquence d'événements, aucune nouvelle production n'est créée, contrairement aux règles spécifiées par le format Efficient XML.

A la suite des étapes 370 et 390, au cours d'une étape 395, on détermine s'il reste des événements à décoder et, si c'est le cas, on retourne à l'étape 300. Sinon, le décodage s'achève.

La Figure 4 décrit, sous forme d'un logigramme, des étapes d'un premier mode de réalisation d'une modification des grammaires en fonction d'un motif. Ces étapes peuvent être appliquées de façon indifférente lors de l'encodage et lors du décodage. Ainsi, l'utilisation d'un motif permet à l'encodeur et au décodeur de conserver des ensembles de grammaires identiques.

Les étapes décrites dans cette figure permettent d'ajouter la description d'un motif dans une grammaire comme une seule production contenant une séquence d'événements : les événements qui composent le motif. Il est à noter que cette méthode ne permet de coder que les motifs ne contenant pas de partie optionnelle, ni plusieurs valeurs possibles pour un même contenu.

Au cours d'une étape 400, on obtient la définition du motif. Puis, au cours d'une étape 410, on crée une production pour représenter le motif. Cette production ne contient initialement aucun événement, et n'a pas encore de priorité associée.

On obtient ensuite le premier événement composant le motif, au cours d'une étape 420. On ajoute cet événement à la production, au cours d'une étape 430, comme le dernier événement dans la liste des événements contenus dans la production.

Ensuite, au cours d'une étape 440, on détermine si le motif contient d'autres événements. Si c'est le cas, on obtient l'événement suivant au cours d'une étape 450, puis on retourne à l'étape 430. Si le résultat de l'étape 440 est négatif, on complète la production avec la grammaire à utiliser après l'utilisation de cette production, au cours d'une étape 460. Pour la réalisation de l'étape

460, deux cas peuvent se présenter : si le dernier événement ajouté à la production correspond à la fin de l'élément que décrit la grammaire, aucune grammaire suivante n'est ajoutée à la production. En effet, dans ce cas, l'élément décrit par la grammaire étant terminé, le grammaire suivante à utiliser correspond à celle en cours d'utilisation pour le parent éventuel de l'élément et n'a donc pas à être précisée dans la production. Dans le cas contraire, la grammaire courante est ajoutée à la production comme grammaire suivante. En effet, dans ce cas, l'élément décrit par la grammaire n'est pas terminé et la grammaire suivante à utiliser est celle décrivant le contenu de l'élément. Une nuance peut être apportée si plusieurs grammaires servent à décrire le contenu de l'élément. Dans un tel cas, la grammaire suivante à utiliser dépendra du contenu du motif et des productions de chacune des grammaires. L'objectif est que la grammaire suivante à utiliser soit la même que celle qui serait à utiliser si l'ensemble des événements codés dans la production l'avait été avec d'autres productions de la grammaire.

Puis, au cours d'une étape 470, on affecte une priorité à la nouvelle production, de préférence une priorité importante ayant un seul niveau, par exemple la priorité « 0 », et on met à jour les priorités des autres productions de la grammaire pour conserver l'unicité des priorités dans la grammaire.

La Figure 5 décrit, sous forme d'un logigramme, des étapes d'un deuxième mode de réalisation d'une modification des grammaires en fonction d'un motif. Ces étapes peuvent être appliquées de façon indifférente lors de l'encodage et lors du décodage. Ainsi, l'utilisation d'un motif permet à l'encodeur et au décodeur de conserver des ensembles de grammaires identiques.

Les étapes décrites dans cette figure permettent d'ajouter la description d'un motif comme un ensemble de nouvelles grammaires, chaque grammaire décrivant l'un des événements qui composent le motif.

Au cours d'une étape 500, on obtient la définition du motif. Le motif correspondant à la description d'un élément ou d'un partie de cet élément, la première grammaire courante utilisée (et modifiée) est la grammaire correspondant à cet élément ou à cette partie de l'élément.

Puis, on obtient le premier événement composant ce motif, au cours d'une étape 510.

Au cours d'une étape 520, on crée une ou plusieurs productions pour représenter cet événement. Dans le cas général, une seule production est créée, contenant l'événement à représenter. Un premier cas particulier est celui où l'événement correspond à un choix entre plusieurs événements (événement « OR »). Dans ce cas, on obtient l'ensemble des événements proposés par le choix et on construit une production pour chacun de ces événements. Un autre cas particulier est celui où l'événement correspond à une séquence d'événements optionnelle. Dans ce cas, le premier événement de cette séquence optionnelle est obtenu et une première production est créée pour représenter ce premier événement. Puis, une deuxième production est créée pour représenter l'événement survenant si la séquence optionnelle n'est pas choisie. Cet événement n'étant pas connu, la production est mise en attente pour être complétée ultérieurement.

Lors de l'étape 520, on détermine, en outre, si l'événement obtenu suit une séquence optionnelle. Si c'est le cas, cet événement est utilisé pour compléter les productions mises en attente. Dans le cas où l'événement obtenu correspond au début d'une nouvelle séquence optionnelle, les productions mises en attente sont dupliquées et l'événement obtenu est utilisé pour compléter les productions dupliquées.

Puis, au cours d'une étape 530, les grammaires sont mises à jour. Si une seule production a été créée, on détermine si elle existe déjà dans la grammaire. Si c'est le cas, la production créée est détruite et la grammaire suivante devient celle de la production existante. Si ce n'est pas le cas, la production est ajoutée à la grammaire avec une priorité importante, par exemple la priorité « 0 », et la grammaire suivante est indéfinie.

Si plusieurs productions ont été créées, on détermine si au moins l'une d'entre elles existe déjà dans la grammaire et, si oui, si chaque production qui existe déjà dans la grammaire fait référence à la même grammaire suivante. Si c'est le cas, les productions qui n'existent pas déjà dans la grammaire sont ajoutées à la grammaire, avec des priorités importantes, et avec comme

grammaire suivante la grammaire suivante commune aux productions qui existaient déjà dans la grammaire. Si ce n'est pas le cas, l'ensemble des productions est ajouté à la grammaire avec une grammaire suivante indéfinie.

Enfin, chacune des productions mises en attente et complétées est traitée. Deux cas peuvent se présenter. Si la grammaire suivante est indéfinie, les productions mises en attente sont ajoutées à la grammaire avec une priorité importante et une grammaire suivante indéfinie. Si la grammaire suivante est définie, pour chaque production en attente, on détermine si cette production existe déjà dans la grammaire avec cette même grammaire suivante. Si c'est le cas, aucun traitement supplémentaire n'est effectué. Si ce n'est pas le cas, la production est ajoutée à la grammaire avec comme grammaire suivante cette grammaire suivante définie.

Puis, au cours d'une étape 540, on détermine s'il reste d'autres événements à traiter dans le motif. Si c'est le cas, on obtient l'événement suivant, au cours d'une étape 550, puis on obtient la grammaire suivante, au cours d'une étape 560. Si la grammaire suivante a été définie à l'étape 530, alors elle devient la grammaire courante (dans laquelle seront ajoutées les prochaines productions créées). Si ce n'est pas le cas, une nouvelle grammaire vide est créée, et l'ensemble des productions créées précédemment ou complétées précédemment sont mises à jour pour indiquer, comme grammaire suivante, cette nouvelle grammaire. Puis, on retourne à l'étape 520.

Si le résultat de l'étape 540 est négatif, c'est-à-dire quand tous les événements composant le motif on été traités, on procède à une étape de finalisation 570. Cette étape 570 consiste à définir la grammaire suivante pour les productions créées ou complétées précédemment et n'ayant pas de grammaire suivante définie. Si le dernier événement ajouté à la production correspond à la fin de l'élément que décrit la première grammaire utilisée, aucune grammaire suivante n'est ajoutée à la production. En effet, dans ce cas, l'élément décrit par la grammaire étant terminé, le grammaire suivante à utiliser correspond à celle en cours d'utilisation pour le parent de l'élément et n'a donc pas à être précisée dans la production. Dans le cas contraire, la grammaire suivante à utiliser dépend du contenu du motif et des productions de chacune

des grammaires. L'objectif est que la grammaire suivante à utiliser soit la même que celle qui serait à utiliser si l'ensemble des événements codés dans la production l'avait été avec d'autres productions de la grammaire.

Pour aider au choix des grammaires à utiliser lors du codage, chaque production ajoutée pour représenter un motif peut être associée à ce motif. Ceci concerne aussi les productions déjà existantes identiques à une production nécessaire pour représenter le motif. Ainsi, quand un ensemble d'items associés à un motif est rencontré, l'encodeur choisit les productions à utiliser en fonction du motif associé à ces items.

On note qu'un troisième mode de réalisation d'une modification des grammaires en fonction d'un motif peut être construit comme composition des deux premiers illustrés en regard des figures 4 et 5. Par exemple, pour une séquence d'événements du motif ne comprenant pas d'options ni de choix, le premier mode de réalisation (voir figure 4) est utilisé pour représenter l'ensemble de la séquence à l'aide d'une seule production. Dans le cas d'une option ou d'un choix, le deuxième mode de réalisation (voir figure 5) est utilisé, pour représenter l'option ou le choix par plusieurs productions.

L'intérêt de ce troisième mode de réalisation est de combiner les avantages des deux premiers modes de réalisation. Le premier mode de réalisation permet en effet de diminuer le nombre de grammaires et de productions créées. Le deuxième mode de réalisation permet plus de possibilités dans les motifs pouvant être intégrés aux grammaires.

On observe que la présente invention ne se limite pas au mode de réalisation décrit et représenté mais s'étend, au contraire, au codage et au décodage de données hiérarchisées, ou « structurées » organisées en une pluralité d'items pouvant êtres décrits par un ensemble d'événements en utilisant une grammaire comprenant des productions.

REVENDICATIONS

1 - Procédé de codage de données hiérarchisées organisées en une pluralité d'items pouvant êtres décrits par un ensemble d'événements en utilisant au moins une grammaire comportant des productions, ledit procédé comportant une étape d'adjonction d'au moins une production à une dite grammaire, caractérisé en ce que ladite étape d'adjonction comporte :

- une étape (210) d'obtention d'un motif structurel représentant une pluralité d'informations au moins structurelles représentatives d'au moins un item à coder ;
- une étape (420 à 450, 510 à 560) d'extraction d'une pluralité d'événements décrivant ledit motif structurel obtenu ;
- une étape (520) de création d'au moins une production comportant, ensemble, la pluralité d'événements extraits ; et
- une étape (530) d'insertion de chaque dite production créée dans une dite grammaire.

2 – Procédé selon la revendication 1, caractérisé en ce que, au cours de l'étape (210) d'obtention d'un motif structurel, ledit motif structurel représente, en outre, des informations de contenu d'au moins un item à coder.

3 – Procédé selon l'une quelconque des revendications 1 ou 2, caractérisé en ce qu'il comporte, en outre, une étape (530) d'association d'un code à chaque production créée.

4 – Procédé selon la revendication 3, caractérisé en ce que, au cours de l'étape (530) d'association d'un code, on affecte à la nouvelle production un code à faible coût de codage.

5 – Procédé selon l'une quelconque des revendications 1 à 4, caractérisé en ce qu'il comporte, en outre, une étape (230) de codage d'au moins un motif structurel obtenu, en mettant en œuvre au moins une dite grammaire.

6 – Procédé selon la revendication 5, caractérisé en ce que, au cours de l'étape (230) de codage d'au moins un motif structurel, au moins un motif structurel est codé en codant :

- un événement de début de description de motif,

- la pluralité des événements extrait décrivant ledit motif, et
- un événement de fin de description de motif.

7 – Procédé selon l'une quelconque des revendications 5 ou 6, caractérisé en ce que, au cours de l'étape (230) de codage d'au moins un motif structurel obtenu, pour au moins un début d'élément contenu dans un motif structurel, on code une information indiquant si le contenu de cet élément est décrit dans le motif.

8 – Procédé selon l'une quelconque des revendications 1 à 7, caractérisé en ce qu'il comporte, en outre, une étape (260, 280) de codage d'au moins un item à coder à l'aide d'un motif structurel le représentant.

9 – Procédé selon l'une quelconque des revendications 1 à 8, caractérisé en ce que, au cours de l'étape (520) de création d'au moins une production, on crée une production comportant au moins deux événements extraits décrivant le motif structurel obtenu.

10 – Procédé selon la revendication 9, caractérisé en ce que, au cours de l'étape (520) de création d'au moins une production, on crée une production comportant un événement décrivant un premier item et au moins un autre événement décrivant un deuxième item contenu dans ledit premier item.

11 – Procédé selon la revendication 10, caractérisé en ce qu'il comporte, pour des items rassemblés en une seule production, une étape (280) de codage conjoint des dits items rassemblés en une seule production.

12 – Procédé selon l'une quelconque des revendications 1 à 11, caractérisé en ce que, au cours de l'étape (520) de création d'au moins une production, on crée une pluralité de productions comportant, chacune, au moins un événement de la pluralité d'événements obtenus décrivant le motif obtenu.

13 – Procédé selon l'une quelconque des revendications 12, caractérisé en ce que, au cours de l'étape (520) de création de la pluralité de productions, une première production de la pluralité de productions est insérée dans une dite grammaire et ladite grammaire est référencée par une deuxième production de la pluralité de productions.

14 – Procédé selon l'une quelconque des revendications 1 à 13, caractérisé en ce qu'il comporte, préalablement à l'étape (530) d'insertion d'au moins une

production créée dans une grammaire, une étape de vérification de la présence de ladite production dans une dite grammaire, l'étape d'insertion n'étant réalisée que si ladite production n'est pas présente dans une dite grammaire.

15 – Procédé selon l'une quelconque des revendications 1 à 14, caractérisé en ce qu'il comporte, en outre, une étape d'obtention d'au moins une grammaire initiale.

16 – Procédé selon l'une quelconque des revendications 1 à 15, caractérisé en ce que les données hiérarchisées sont exprimées sous la forme d'un document XML.

17 - Procédé de décodage de données hiérarchisées organisées en une pluralité d'items pouvant êtres décrits par un ensemble d'événements en utilisant au moins une grammaire comportant des productions, ledit procédé comportant une étape d'adjonction d'au moins une production à une dite grammaire, caractérisé en ce que ladite étape d'adjonction comporte :

- une étape d'obtention d'un motif structurel représentant une pluralité d'informations au moins structurelles représentatives d'au moins un item à décoder ;
- une étape d'extraction d'une pluralité d'événements décrivant ledit motif structurel obtenu ;
- une étape de création d'au moins une production comportant, ensemble, la pluralité d'événements extraits ; et
- une étape d'insertion de chaque dite production créée dans une dite grammaire.

18 – Procédé selon la revendication 17, caractérisé en ce qu'il comporte, en outre, une étape d'association d'un code à chaque production créée.

19 – Procédé selon l'une quelconque des revendications 17 ou 18, caractérisé en ce qu'il comporte, en outre, une étape de décodage d'au moins un motif structurel obtenu, en mettant en œuvre au moins une dite grammaire.

20 – Dispositif de codage de données hiérarchisées organisées en une pluralité d'items pouvant êtres décrits par un ensemble d'événements en utilisant au moins une grammaire comportant des productions, ledit dispositif comportant

un moyen d'adjonction d'au moins une production à une dite grammaire, caractérisé en ce que ledit moyen d'adjonction comporte :

- un moyen d'obtention d'un motif structurel représentant une pluralité d'informations au moins structurelles représentatives d'au moins un item à coder ;
- un moyen d'extraction d'une pluralité d'événements décrivant ledit motif structurel obtenu ;
- un moyen de création d'au moins une production comportant, ensemble, la pluralité d'événements extraits ; et
- un moyen d'insertion de chaque dite production créée dans une dite grammaire.

21 - Dispositif de décodage de données hiérarchisées organisées en une pluralité d'items pouvant êtres décrits par un ensemble d'événements en utilisant au moins une grammaire comportant des productions, ledit dispositif comportant un moyen d'adjonction d'au moins une production à une dite grammaire, caractérisé en ce que ledit moyen d'adjonction comporte, en outre :

- un moyen d'obtention d'un motif structurel représentant une pluralité d'informations au moins structurelles représentatives d'au moins un item à décoder ;
- un moyen d'extraction d'une pluralité d'événements décrivant ledit motif structurel obtenu ;
- un moyen de création d'au moins une production comportant, ensemble, la pluralité d'événements extraits ; et
- un moyen d'insertion de chaque dite production créée dans une dite grammaire.

22 - Programme d'ordinateur chargeable dans un système informatique, ledit programme contenant des instructions permettant la mise en œuvre du procédé de codage selon l'une quelconque des revendications 1 à 16 et/ou du procédé de décodage selon l'une quelconque des revendications 17 à 19, lorsque ce programme est chargé et exécuté par un système informatique.

23 - Support d'informations lisibles par un ordinateur ou un microprocesseur, amovible ou non, conservant des instructions d'un programme informatique,

caractérisé en ce qu'il permet la mise en œuvre du procédé de codage selon l'une quelconque des revendications 1 à 16 et/ou du procédé de décodage selon l'une quelconque des revendications 17 à 19.

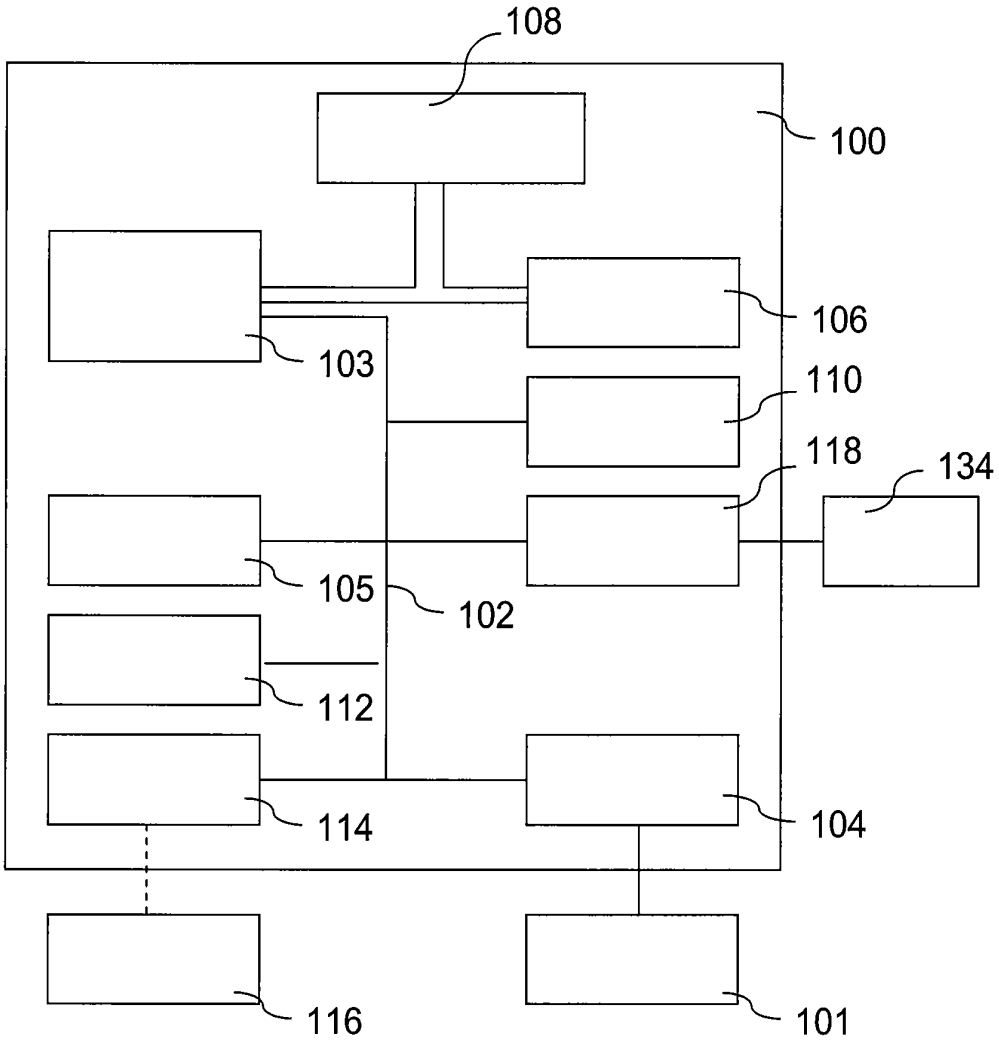


Figure 1

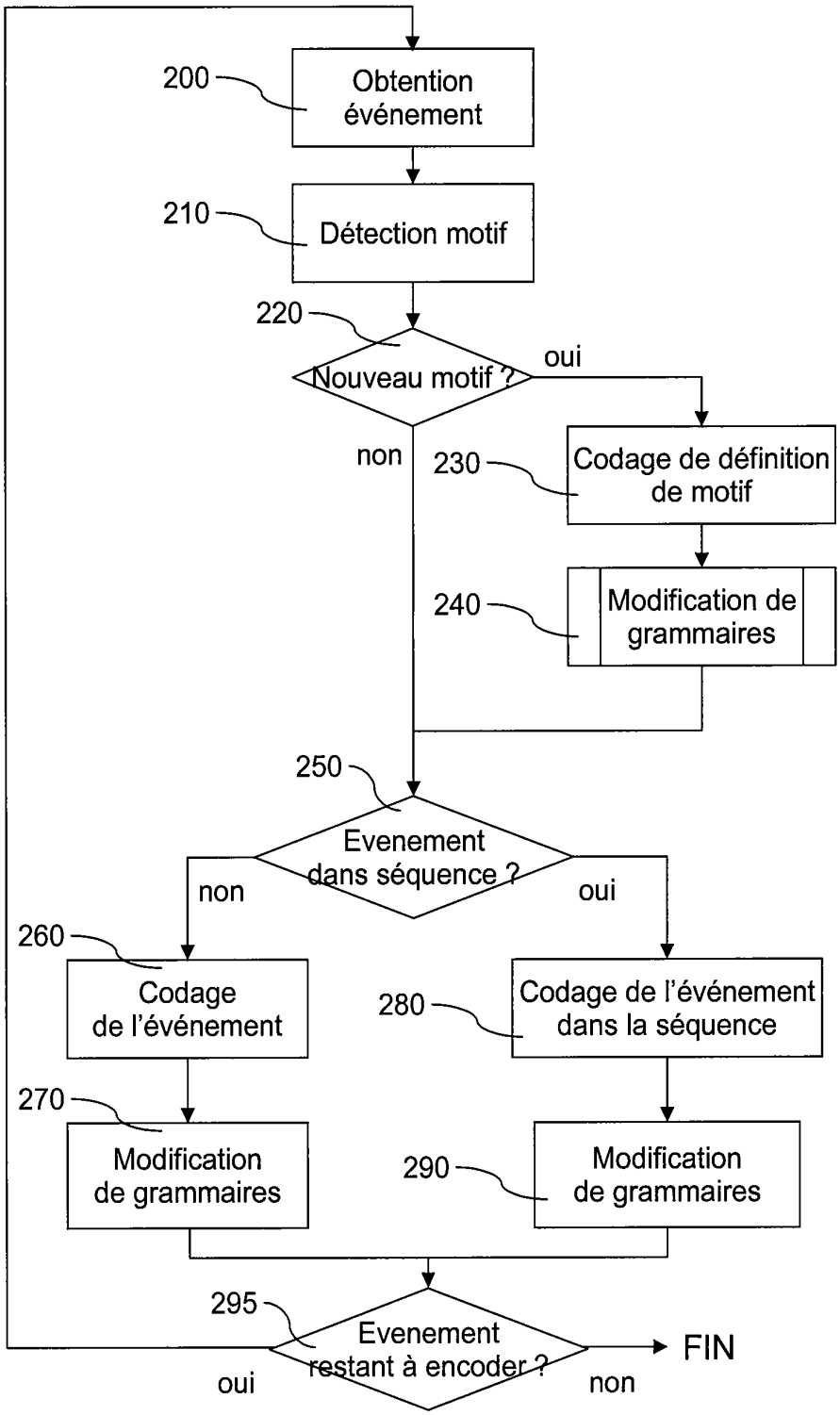

200
Obtention événement
210
Détection motif
220
Nouveau motif ?
oui
non
230
Codage de définition de motif
240
Modification de grammaires
250
Evenement dans séquence ?
non
oui
260
Codage de l'événement
270
Modification de grammaires
280
Codage de l'événement dans la séquence
290
Modification de grammaires
295
Evenement restant à encoder ?
oui
non
FIN

Figure 2

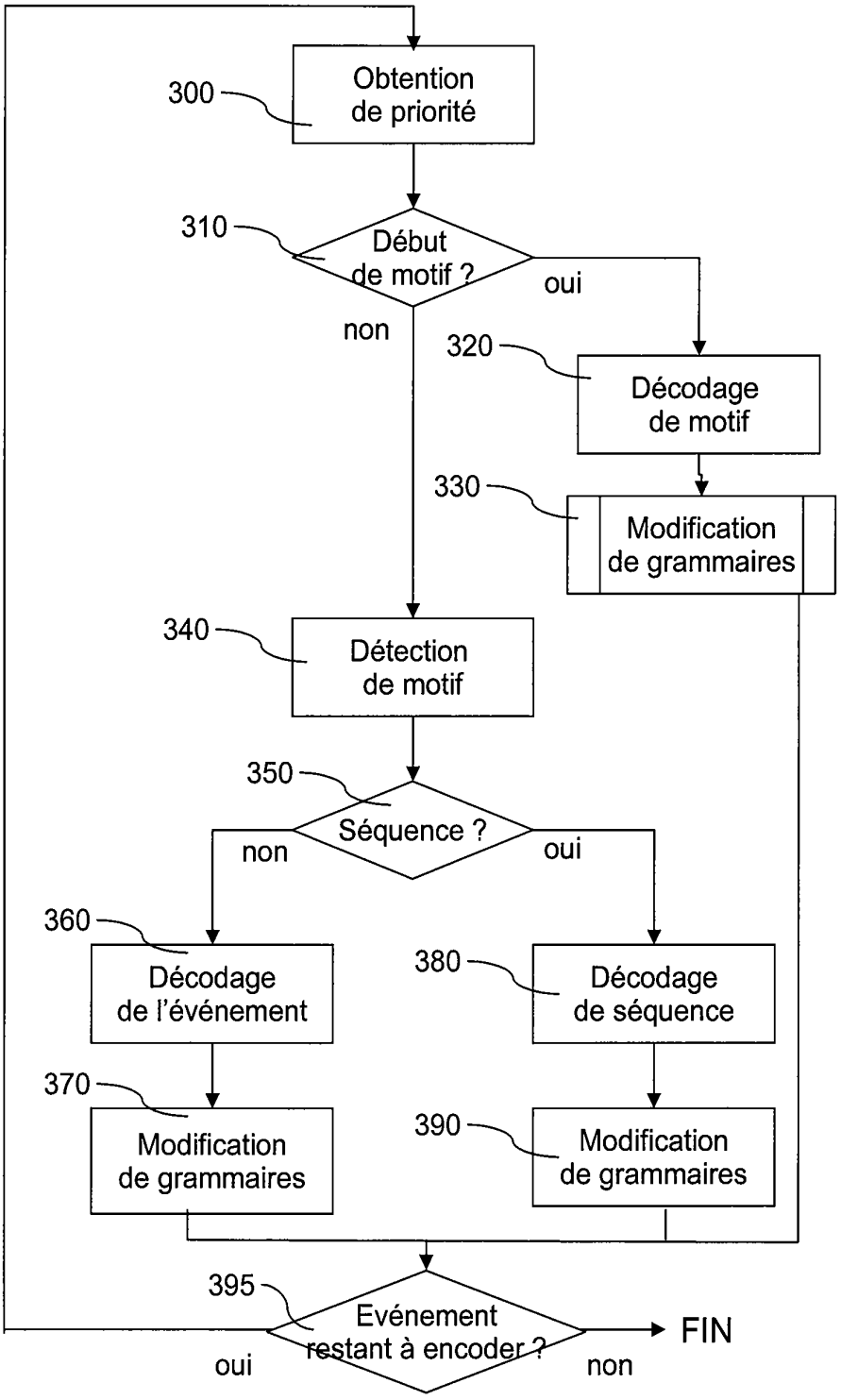

300
Obtention de priorité
310
Début de motif ?
oui
non
320
Décodage de motif
330
Modification de grammaires
340
Détection de motif
350
Séquence ?
non
oui
360
Décodage de l'événement
370
Modification de grammaires
380
Décodage de séquence
390
Modification de grammaires
395
Evénement restant à encoder ?
oui
non
FIN

Figure 3

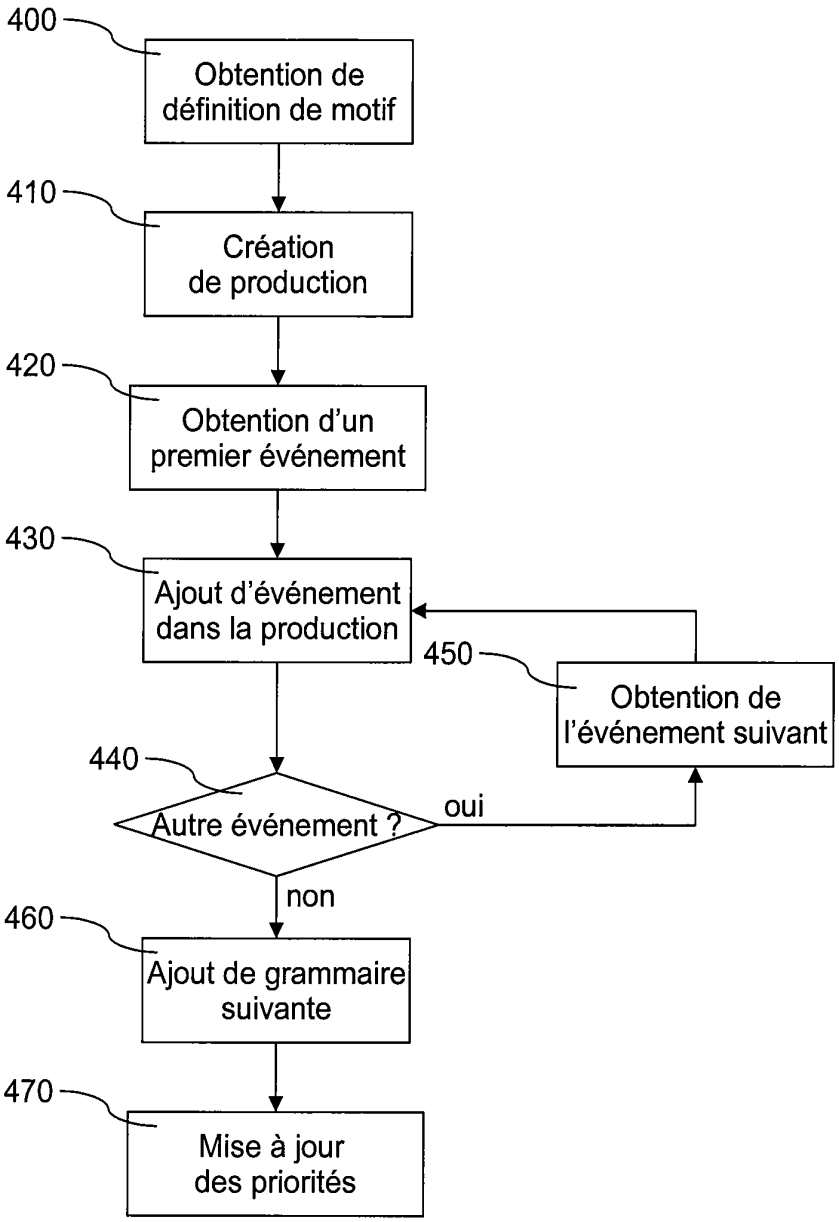

400
Obtention de définition de motif
410
Création de production
420
Obtention d'un premier événement
430
Ajout d'événement dans la production
450
Obtention de l'événement suivant
440
Autre événement ?
oui
non
460
Ajout de grammaire suivante
470
Mise à jour des priorités

Figure 4

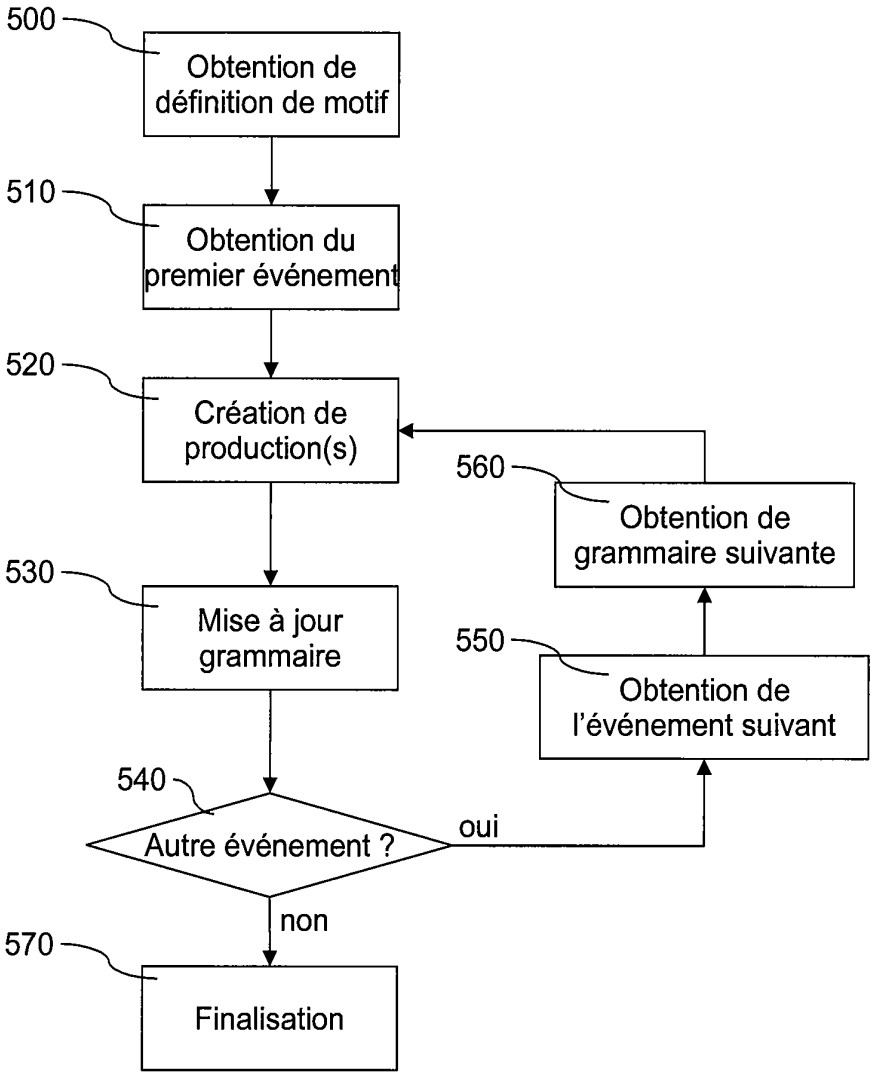

500
Obtention de définition de motif
510
Obtention du premier événement
520
Création de production(s)
530
Mise à jour grammaire
540
Autre événement ?
oui
non
550
Obtention de l'événement suivant
560
Obtention de grammaire suivante
570
Finalisation

Figure 5

RÉPUBLIQUE FRANÇAISE

2913275

INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

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

N° d'enregistrement national

FA 694106
FR 0753632

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	SCHNEIDER J C: "Theory, Benefits and Requirements for Efficient Encoding of XML Documents" ONLINE AGILE DELTA W3C SUBMISSION, août 2003 (2003-08), XP002458724 Extrait de l'Internet: URL:http://www.w3.org/2003/08/binary-interchange-workshop/30-agiledelta-Efficient-updated.html> [extrait le 2007-11-15] * le document en entier *	1,17, 20-23	G06F17/22 H03M7/00
A	WHITE G ET AL: "Efficient XML Interchange Measurements Note" ONLINE W3C WORKING DRAFT, 18 juillet 2006 (2006-07-18), XP002458707 Extrait de l'Internet: URL:http://www.w3.org/TR/2006/WD-exi-measurements-20060718/> [extrait le 2007-11-15] * section 5.7 *	1,17, 20-23	
T	SCHNEIDER J ET AL: "Efficient XML Interchange (EXI) Format 1.0" ONNLINE W3C WORKING DRAFT, 16 juillet 2007 (2007-07-16), XP002458708 Extrait de l'Internet: URL:http://www.w3.org/TR/2007/WD-exi-20070716/> [extrait le 2007-11-15] * le document en entier *	1,17, 20-23	DOMAINES TECHNIQUES RECHERCHÉS (IPC) G06F
A	ADIEGO J ET AL: "Lempel-Ziv compression of highly structured documents" JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE AND TECHNOLOGY, WILEY & SONS, NEW YORK, NY, US, vol. 58, no. 4, 25 janvier 2007 (2007-01-25), pages 461-478, XP002455183 ISSN: 1532-2882 * abrégé *	1,17, 20-23	

Date d'achèvement de la recherche	Examinateur
15 novembre 2007	Stauch, Marc

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)