(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 931 275**

(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **08 53119**

(51) Int Cl[8] : ***G 06 F 17/50*** *(2006.01)*

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 14.05.08.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 20.11.09 Bulletin 09/47.

(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) :** *AIRBUS FRANCE Société par actions simplifiée* — FR et *AIRBUS UK* — GB.

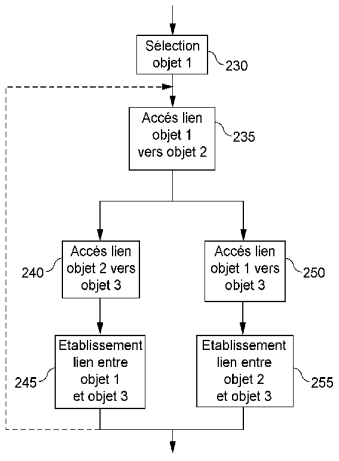
(72) **Inventeur(s) :** MARQUEZ BERNARD, CHEVALIER THIERRY et TURSI STEFANO.

(73) **Titulaire(s) :** AIRBUS FRANCE Société par actions simplifiée, AIRBUS UK.

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

(54) PROCEDE POUR LA TRACABILITE DE DONNEES DANS UN ATELIER ORIENTE SERVICE COLLABORATIF.

(57) L'invention a notamment pour objet un procédé de gestion de données dans un atelier orienté service collaboratif adapté à traiter des objets associés à des données représentatives de données réelles ou de processus. Après avoir sélectionné (230) au moins un premier objet comprenant au moins un lien, appelé premier lien, vers un second objet, distinct dudit premier objet, au moins un lien vers un troisième objet, appelé second lien, est accédé (240, 245), ledit second lien étant compris dans l'un desdits premier et second objets, ledit troisième objet étant distinct desdits premier et second objets. Un lien est ensuite établi (245, 255) entre les données associées audit troisième objet et l'autre desdits premier et troisième objets.



FR 2 931 275 - A1

La présente invention concerne les architectures informatiques pour le travail collaboratif et plus particulièrement un procédé pour la traçabilité de données dans un atelier orienté service collaboratif.

L'optimisation des phases d'étude et de conception d'objets tels que des véhicules dans le but de réduire les coûts et les temps de développement nécessite généralement la mise en œuvre d'un environnement informatique spécifique.

Par exemple, l'étude aérodynamique d'un aéronef fait intervenir de nombreux spécialistes, travaillant sur des aspects différents des mêmes données, ces aspects pouvant être liés ou non les uns aux autres. Ces spécialistes utilisent généralement des outils informatiques différents ainsi que des machines différentes pour l'exécution des processus mis en œuvre. De plus, ces spécialistes peuvent être situés, géographiquement, à des endroits distincts.

Il s'avère donc avantageux de mettre en œuvre un environnement basé sur une architecture ouverte permettant le partage des données et des ressources informatiques, notamment des ressources de calcul. Un tel environnement doit de préférence s'adapter à un ensemble de machines et de systèmes d'exploitation hétérogènes.

Il est également avantageux d'utiliser une interface permettant d'accéder à différents outils existants distincts. Une telle interface commune doit notamment être adaptée à gérer les données devant être échangées entre ces outils ainsi qu'à permettre une sélection et un accès faciles aux données.

De plus, l'environnement doit de préférence permettre une adaptation facile aux besoins futurs tels que l'intégration de nouveaux outils, la gestion de nouveaux types de données et la production de nouveaux types de résultats.

Par ailleurs, il existe un besoin de traçabilité des données pour déterminer l'origine des données produites ainsi que leur cycle de vie au cours des différentes modifications et utilisations dans des processus de calcul ou autres.

Ces problèmes sont partiellement résolus les uns indépendamment des autres.

Il existe par exemple des applications logicielles telles que SynfiniWay développé par la société Fujitsu (SynfiniWay et Fujitsu sont des marques) qui permettent l'optimisation de l'exécution de tâches dans un environnement hétérogène. L'application SynfiniWay permet notamment de virtualiser les ressources globales et de présenter les applications sous forme de services reliés par des séquences et des dépendances de données pour automatiser les processus informatiques.

Il existe par ailleurs des langages de script tels que Python et Java (Python et Java sont des marques) qui permettent d'automatiser certaines tâches et de faire appel à des outils existants. Python est un langage de programmation de haut niveau, interprété et orienté objet. Il peut s'utiliser dans de nombreux contextes et s'adapter à de nombreuses utilisations à l'aide de bibliothèques spécialisées.

La traçabilité des données est quant à elle généralement déterminée par les processus ayant conduit aux données. Par exemple, dans les applications de type PDM (sigle de *Product Data Management* en terminologie anglo-saxonne), l'utilisateur doit suivre des processus prédéterminés, sans pouvoir y déroger. L'origine d'une donnée est ainsi déterminée par le processus qui a permis d'obtenir la donnée.

Cependant, il n'existe pas d'environnement centralisé et optimisé pour l'étude et la conception de systèmes dans un environnement collaboratif ouvert hétérogène.

L'invention permet de résoudre au moins un des problèmes exposés précédemment.

L'invention a ainsi pour objet un procédé de gestion de données dans un atelier orienté service collaboratif adapté à traiter des objets associés à

des données représentatives de données réelles ou de processus, ce procédé comprenant les étapes suivantes,

- sélection d'au moins un premier objet comprenant au moins un lien, appelé premier lien, vers un second objet, distinct dudit premier objet ;
- accès à au moins un lien vers un troisième objet, appelé second lien, ledit second lien étant compris dans l'un desdits premier et second objets, ledit troisième objet étant distinct desdits premier et second objets ; et,
- établissement d'un lien entre les données associées audit troisième objet et l'autre desdits premier et deuxième objets.

Le procédé selon l'invention permet ainsi de gérer des données indépendamment de leurs caractéristiques propres telles que leur nature, leur lieu de stockage et leur taille pour permettre une gestion globale de celles-ci et permettre de lier des objets entre eux et, ainsi, d'offrir les moyens de suivre une relation de proche en proche entre plusieurs objets.

Selon un mode de réalisation particulier, les données associées audit premier objet sont représentatives d'un processus. Toujours selon un mode de réalisation particulier, les données associées audit second objet sont au moins partiellement obtenues à partir dudit processus ou ledit processus est appliqué sur au moins une partie desdites données associées audit second objet.

De façon avantageuse, le procédé comprend en outre une étape de construction d'un arbre de données selon lesdits premier et second liens permettant d'établir un lien entre lesdits premier et troisième objets ou entre lesdits second et troisième objets.

Lesdites étapes d'accès à au moins un lien et d'établissement d'un lien sont, de préférence, répétées de façon récursive pour établir un lien entre des données associées à au moins deux objets distincts.

Selon un mode de réalisation particulier, au moins l'un desdits liens contenu dans l'un desdits objets est mémorisé dans une table de liens pour faciliter la gestion des liens et permettre une construction efficace d'arbres de données.

Toujours selon un mode de réalisation particulier, ledit processus fait appel à au moins une fonction extérieure audit atelier orienté service collaboratif

afin d'utiliser des fonctions existantes et limiter les développements logiciels. Le procédé selon l'invention permet ainsi d'éviter de maintenir plusieurs versions similaires de modules logiciels.

Toujours selon un mode de réalisation particulier, lesdits objets sont mémorisés dans une base de données centralisée et lesdites données associées auxdits objets sont mémorisées dans des dispositifs distants pour optimiser la gestion des ressources.

L'invention a également pour objet un programme d'ordinateur comprenant des instructions adaptées à la mise en œuvre de chacune des étapes du procédé décrit précédemment.

D'autres avantages, buts et caractéristiques de la présente invention ressortent de la description détaillée qui suit, faite à titre d'exemple non limitatif, au regard des dessins annexés dans lesquels :

- la figure 1 représente schématiquement un exemple d'architecture d'un espace de travail pour le partage des données et des ressources selon l'invention ;

- la figure 2, comprenant les figures 2a et 2b, illustre un schéma fonctionnel simplifié du gestionnaire de données présenté sur la figure 1 et un exemple d'algorithme permettant d'établir un arbre de données, respectivement ;

- la figure 3 représente schématiquement les éléments fonctionnels du moteur d'exécution illustré sur la figure 1, ainsi que les informations transmises entre ces éléments, pour l'exécution d'un processus basé sur une représentation de type Python ;

- la figure 4 illustre l'atelier orienté service collaboratif selon l'invention, d'un point de vue applicatif, permettant l'exécution de code exécutable issu de développements externes et de développements spécifiques ;

- la figure 5 illustre un exemple d'encapsulation d'une fonction externe de type modèle ModelCenter dans un modèle de processus ;

- la figure 6 illustre de façon synthétique un exemple de schéma de type XML pour l'analyse d'un modèle de processus mémorisé sous forme de description de type XML ;

- la figure 7 illustre un exemple d'algorithme pour l'exécution de processus unitaires de modèles de processus instanciés ;

- la figure 8 illustre un exemple de traitement de données pouvant être effectuée à partir d'un atelier orienté service collaboratif selon l'invention ; et,

- la figure 9 illustre un exemple de dispositif adapté à mettre en œuvre une partie de l'invention.

La figure 1 représente schématiquement un exemple d'une architecture 100 d'un espace de travail pour le partage des données et des ressources selon l'invention, appelé aussi atelier orienté service collaboratif. Comme illustré, cette architecture est réalisée autour de quatre modules principaux : le module 105 qui a pour objet l'accès aux données, le module 110 qui est un gestionnaire de données, le module 115 qui est un moteur d'exécution et le module 120 qui a pour objet la gestion des processus.

Le module 105 d'accès aux données est relié au module 110 de gestion des données ainsi qu'à une base de données centralisée 125, par exemple une base de données de type Oracle (Oracle est une marque). Le module 110 de gestion des données est relié à la base de données centralisée 125, au moteur d'exécution 115 et à un ensemble 130 représentant des zones de stockage distribuées pouvant être organisées de différentes façons, par exemple sous forme de fichiers ou de bases de données. Le moteur d'exécution 115 est lui-même relié à l'ensemble 130 de zones de stockage et au module 120 de gestion des processus.

Bien que cette architecture soit particulièrement adaptée à la conception d'objets complexes tels que les aéronefs, la description suivante est basée sur un exemple simple pour des raisons de clarté et de concision. Cet exemple est donné dans un but illustratif.

L'architecture 100 est adaptée à manipuler des objets informatiques représentant des données dans le sens large du terme, ces données pouvant

être représentatives d'un objet réel, par exemple d'un aéronef, ou d'un processus tel qu'un processus de calcul de traînée aérodynamique. De façon simplifiée, l'atelier orienté service collaboratif travaille sur des données représentées par des objets informatiques basés eux-mêmes sur des méta-données qui représentent des données caractéristiques.

Les données, appelées *framework data* en terminologie anglo-saxonne ou FD, sont donc des données physiques mémorisées, par exemple, sous forme de fichiers dans une zone de stockage 130. Ces données caractérisent un objet réel, par exemple le maillage ou la structure d'une chaise, ou un processus, par exemple, les étapes d'un procédé de déménagement. La taille de ces données n'est pas limitée, elle peut être de plusieurs centaines de méga octets. Toutes les données utilisées peuvent être mémorisées dans des emplacements physiques distincts.

Les objets, appelés *framework object* en terminologie anglo-saxonne ou FO, représentent des objets ayant une signification particulière au sens applicatif. Comme indiqué précédemment, ces objets sont utilisés pour représenter tout ce qui peut être manipulé par l'atelier orienté service collaboratif. Les objets sont basés sur des ensembles de méta-données pouvant être comprises comme des éléments de base ou des éléments caractéristiques. Un ensemble de méta-données est ainsi associé à chaque objet. Chaque type d'ensemble de méta-données peut être associé à plusieurs objets. La qualification d'un type d'ensemble de méta-données permet de former un objet. Si, par exemple, un type d'ensemble de méta-données est associé à une chaise, une qualification de la chaise formant un objet peut être une chaise d'enfant.

Dans un souci de clarté, les implémentations informatiques des objets, appelées *framework object proxies* en terminologie anglo-saxonne ou FO *proxies*, sont ici assimilées aux objets eux-mêmes.

Parmi les objets figurent les modèles de processus instanciés, appelés *instanciated process templates* en terminologie anglo-saxonne ou IPT. Les modèles de processus instanciés sont des instances de modèles de processus, c'est-à-dire des modèles de processus appliqués à un ou plusieurs

objets. Les modèles de processus, appelés *process templates* en terminologie anglo-saxonne ou PT, sont génériques. Il s'agit par exemple d'un processus de construction d'un salon à partir de plusieurs chaises et d'une table, d'un processus de rénovation d'une maison ou d'un processus de déménagement. Les modèles de processus instanciés sont représentés sous forme d'objets et peuvent être manipulés en tant que tels dans l'atelier orienté service collaboratif.

Les méta-données, appelées *framework meta-data* en terminologie anglo-saxonne ou FMD, correspondent à des informations particulières des données. Elles permettent de caractériser des données afin de faciliter leur manipulation. Une donnée peut être associée à plusieurs ensembles de méta-données. Les méta-données sont avantageusement mémorisées dans la base de données centralisée 125 sous forme de tableaux ou de fichiers XML (sigle d'*Extensible Markup Language* en terminologie anglo-saxonne). Un type est ici associé à chaque ensemble de méta-données. Les méta-données comprennent des informations génériques telles qu'un identifiant, une date de création et un nom de créateur et des informations spécifiques, propre à la donnée associée, par exemple le matériau, la couleur ou le poids d'une chaise. Les méta-données comprennent également des liens vers des objets qui permettent notamment de déterminer comment les données ont été obtenues et/ou comment elles ont été utilisées. Ces liens permettent de définir des modèles de données, c'est-à-dire des arborescences de données.

La consistance d'un objet est vérifiée à travers les liens définis dans l'ensemble des méta-données qui lui est associé. Par exemple, un modèle de données de table peut être déterminé de telle sorte qu'une table soit liée à une chaise. Si un objet de type table est effectivement associé à un objet de type chaise, alors l'objet de type table est dit consistant au sens du modèle de données défini à travers ses liens.

Les méta-données sont par exemple décrites à l'aide d'un langage de type XML selon la structure présentée en annexe sous la référence « extrait de code 1 ».

La partie générique des méta-données est commune à toutes les données, elle est donc définie lors de la mise en œuvre de l'atelier orienté service collaboratif. Même si cette partie n'a, a priori, pas de raison d'être modifiée, sa définition peut néanmoins évoluer selon les besoins. Cependant, son évolution a une influence sur l'ensemble des objets gérés dans l'atelier orienté service collaboratif.

La partie spécifique des méta-données est propre à chaque type de données et, par conséquent, à chaque type d'ensemble de méta-données. La définition de cette partie est réalisée par les utilisateurs de l'atelier orienté service collaboratif. Les informations de la partie spécifique des méta-données sont déterminées par les opérations d'extraction automatique devant être menées sur les données.

Les méta-données génériques et spécifiques sont avantageusement utilisées pour la recherche et le tri des données contenues dans l'atelier orienté service collaboratif.

L'exemple donné en annexe sous la référence « extrait de code 2 » illustre la description de type XML d'un type de méta-données utilisé pour décrire une chaise.

Comme décrit précédemment, les méta-données comprennent avantageusement une partie générique et une partie spécifique. La partie générique comprend notamment, ici, les éléments suivants,

- une clé permettant de gérer le contrôle d'accès à la donnée ;
- une description du type d'objet ;
- un commentaire décrivant la donnée ;
- une date de création ;
- un identifiant de la donnée qui peut être un clé ou une URL permettant d'accéder à la donnée ;
- un identifiant unique de la donnée ;
- une information concernant l'utilisateur qui a créé la donnée ;
- l'emplacement physique de la donnée avec sa situation géographique ; et,

- son état permettant de savoir si une modification est en cours et par qui.

La seconde partie des méta-données, spécifique à l'objet manipulé, ici une chaise, comprend dans cet exemple les attributs suivants,

- le nombre de pieds ;
- l'épaisseur ;
- la couleur ;
- la matière ; et,
- le poids.

Les parties générique et spécifique peuvent naturellement comprendre d'autres types d'informations.

L'identifiant ou l'adresse d'un ensemble de méta-données permet de retrouver les données auxquelles il est associé.

La qualification d'un ensemble de méta-données représentant une chaise peut être, en particulier, « enfant » ou « adulte », permettant d'aboutir à deux objets différents.

De la même façon, il est possible de définir des méta-données et des objets pour des tables. Lors de l'importation d'un objet définissant une table, il est possible de définir, au niveau d'un modèle de données, que cet objet n'est consistant que s'il est relié à au moins un objet de type chaise. Une telle relation entre objets est définie par les liens définis dans les méta-données. Ces liens peuvent aussi être représentés au format XML ou dans un tableau. Lors de l'importation d'un objet de type table, des liens de type modèle de données entre l'objet de type table et les objets de type chaise doivent être posés afin de rendre consistant cet objet avec le modèle de données ainsi défini.

Alternativement, les méta-données peuvent être représentées sous forme de tables. Par exemple, une table peut être associée à chaque type de donnée. Les colonnes de la table représentent les méta-données tandis que chacune de ses lignes représente un objet. La table 1 figurant en annexe illustre un exemple simplifié de représentation d'un type d'ensemble de méta-données pouvant être associées à des données de type chaise. Il convient de remarquer ici que les liens ne sont pas inclus dans la table représentant les

méta-données. Selon cet exemple, les liens sont mémorisés dans une table de liens.

Une table de liens peut être représentée, par exemple, sous la forme du tableau donné en annexe sous la référence table 2 où chaque colonne représente un couple d'identifiants d'objets entre lesquels un lien est établi.

Ainsi, les informations mémorisées dans les tables des méta-données et dans la table de liens permettent de manipuler toutes les données, actuelles et futures, de l'atelier orienté service collaboratif ainsi que de construire des modèles de données.

Il peut être noté ici que la représentation des méta-données sous forme de tables est équivalente à une représentation XML et qu'il est possible d'utiliser un outil de conversion pour passer d'une représentation à une autre.

La représentation des données caractérisant des objets physiques ou des processus est ainsi réalisée à plusieurs niveaux :

- les données brutes ;
- les méta-données qui représentent des sous-ensembles des données brutes, extraits de celles-ci, qui permettent de caractériser les données brutes ;
- les objets, basés sur les valeurs des méta-données, qui sont manipulés par les modules applicatifs ; et,
- les objets consistants qui sont des objets liés à d'autres objets.

Il convient de remarquer ici que selon cette structure, il n'est pas nécessaire de mettre en œuvre des procédés de synchronisation dans l'atelier orienté service collaboratif.

Le module 105 d'accès aux données permet en particulier à un utilisateur d'accéder de façon sélective et centralisée aux méta-données mémorisées dans la base de données centralisée 125 afin notamment de visualiser ces méta-données, de fabriquer des vues sur ces méta-données permettant un tri efficace sur l'ensemble des données accessibles dans l'atelier de service collaboratif ainsi que, éventuellement, de suivre les liens unissant certaines de ces données.

Le module 105 permet également d'enregistrer des méta-données dans la base de données centralisée 125. Il est ainsi possible, pour un utilisateur, d'entrer des méta-données à travers une interface homme-machine (IHM) pour spécifier des informations qui ne figureraient pas dans les données mémorisées dans les zones de stockage 130. Pour accéder et enregistrer des méta-données, l'interface utilisateur peut utiliser, en particulier, des requêtes de type SQL (sigle de *Structured Query Language* en terminologie anglo-saxonne). Cet enregistrement manuel de méta-données s'avère intéressant pour l'ajout d'informations spécifiques ne pouvant pas faire l'objet d'extraction automatique à partir de la donnée lors de l'importation de l'objet dans l'atelier.

De même, le module 105 d'accès aux données permet à un utilisateur d'accéder de façon sélective et centralisée aux données mémorisées dans les zones de stockage 130 à travers le module 110 de gestion des données ainsi que d'enregistrer des données. Pour accéder et enregistrer des données, l'interface utilisateur utilise avantageusement des API (sigle de *Application Programming Interface* en terminologie anglo-saxonne) spécifiques et des services web.

Le gestionnaire de données 110 est utilisé comme interface entre le module 105 d'accès aux données, la base de données centralisée 125, les zones de stockage 130 et le moteur d'exécution 115 adapté notamment à appliquer des processus sur des données. Le gestionnaire de données 110 permet également de contrôler les droits d'accès des utilisateurs.

Selon un mode de réalisation particulier, le gestionnaire de données 110 est composé d'une partie client et d'une partie serveur. La partie client peut être implémentée dans un langage de type Python pour permettre d'accéder aux services web accessibles dans la partie serveur. Toutes les fonctions standard de gestion de données telles que l'importation, l'exportation, la publication, la collaboration et la gestion de droits sont de préférence disponibles. Cette liste est non exhaustive. L'ensemble des fonctions utilisées au cours du cycle de vie de la donnée est disponible à travers cet API

La partie serveur peut être implémentée, par exemple, en langage de type Java ou Python pour permettre l'interfaçage avec les différentes bases de

données et zones de stockage. Cette partie prend aussi avantageusement en charge les mécanismes d'extraction des méta-données à partir des données mémorisées dans les zones de stockage distribuées 130 vers la base de données centralisée 125 contenant les méta-données.

Le moteur d'exécution 115 permet l'exécution de processus par l'intermédiaire de modèles de processus basés, par exemple, sur des applicatifs métiers. Une interface entre le moteur d'exécution 115 et les zones de stockage 130 permet un accès direct ou via une API aux données lors de l'exécution des processus. Le moteur d'exécution 115 est de préférence lié à une grille de calcul, par exemple du type SynfiniWay, qui gère la soumission des différents processus sur une infrastructure distribuée ainsi que le déplacement des données durant l'exécution.

Le moteur d'exécution 115 comprend également, de préférence, une interface avec des applications utilisant des flux de travaux (*workflows* en terminologie anglo-saxonne) comme les applications ModelCenter (ModelCenter est une marque) ou SynfiniWay, c'est-à-dire des moteurs d'exécution ou d'enchaînement de tâches.

Le gestionnaire de processus 120 permet la création de flux de travaux de hauts niveaux, appelés processus composites, qui encapsulent des traitements unitaires. Ces flux de travaux peuvent faire l'objet d'études paramétriques et de sensibilités aux paramètres et peuvent permettre la création de modèles de comportement. Ces fonctionnalités peuvent ainsi être utilisées dans le cadre d'études multidisciplinaires d'optimisation.

Ainsi, le traitement de processus non unitaires, c'est-à-dire de processus faisant appels à plusieurs fonctions différentes, est géré par le gestionnaire de processus pour être, par exemple, décomposés en processus unitaires qui sont alors exécutés par le moteur d'exécution.

La figure 2a est un schéma fonctionnel simplifié du gestionnaire de données 110.

A l'aide, par exemple, d'une interface fichier ou d'une interface de type Corba (acronyme de *Common Object Request Broker Architecture* en

terminologie anglo-saxonne, Corba est une marque), des données sont créées dans les différentes bases de données (non représenté).

Un mécanisme 200 d'extraction, de transformation et de chargement, appelé ETL (sigle de *Extract, Transform and Load* en terminologie anglo-saxonne), est alors mis en œuvre afin d'extraire les méta-données des données mémorisées dans les zones de stockage 130. Elles sont stockées dans la base de données centralisée 125. Cette base de données collecte ainsi les informations des données physiques créées dans les différentes zones de stockage distribuées 130.

Il convient de remarquer ici que le mécanisme d'extraction est mis en œuvre lorsque les données sont créées, modifiées ou supprimées. Il est propre à chaque type de données et de méta-données.

Une fonction de qualification 205 permet de qualifier les méta-données extraites pour former des objets pouvant être mémorisés dans la base de données centralisée 125. Suite à la qualification, une vérification est faite sur la consistance de l'objet. Cette consistance peut être assurée à travers des liens mémorisés avec les méta-données ou non, ces liens pouvant être de différentes natures telles que la version, la configuration, le modèle de données et l'utilisateur.

Un ensemble de liens entre objets, définissant un modèle de données, peut être utilisé pour déterminer des liens entre plusieurs objets. Les liens sont associés aux méta-données. Comme décrit précédemment, ils sont mémorisés avec ces dernières ou de façon distincte, par exemple dans une table de liens mémorisée dans la base de données centralisée 125.

Les liens permettent d'établir ultérieurement la traçabilité des données, c'est-à-dire la traçabilité entre les différentes données produites ou utilisées lors de l'exécution de processus. Il existe plusieurs types de liens, notamment les suivants,

- les liens utilisateurs déterminés manuellement par l'utilisateur entre deux données ;

- les liens modèles permettant de définir un modèle de donnée. Ces liens sont déterminés de façon générique lors de la création d'un modèle de données ;

- les liens de configuration permettant d'établir un lien entre un ensemble d'objets pour les gérer en configuration, c'est-à-dire pour permettre l'application de fonctions identiques telles que des fonctions d'exportation, d'importation et de changement de droit ;

- les liens de version qui permettent de gérer un même objet selon différentes versions ; et,

- les liens de production qui permettent d'établir un lien d'exécution entre un ensemble d'objets. Ce type de lien est créé automatiquement lors de la phase de publication des données au cours ou à la fin de l'exécution d'un processus instantié (IPT).

L'ensemble de ces liens posés deux à deux permet d'obtenir la traçabilité portée par la donnée : les données contiennent tous les historiques notamment en ce qui concerne les exécutions de processus, les versions, les configurations, les utilisateurs et les modèles.

Il est ainsi possible de construire des arbres complets d'objets reliés entre eux par des méta-données et des liens spécifiques. Dans l'exemple de la chaise et la table, un arbre permettant de connaître les méta-données (couleur et localisation) de la table vérifie le modèle de donnée la liant aux chaises ayant comme méta-données une couleur spécifique. Ainsi, de proche en proche, il est possible de reconstruire des arbres complets d'obtention de données.

Il convient de remarquer ici que les modèles de données, c'est-à-dire les arborescences de données, sont spécifiques aux données manipulées dans l'atelier orienté service collaboratif. Cependant, si les modèles de données peuvent être dérivés des liens entre les objets, ils n'existent pas en tant que tels dans l'atelier orienté service collaboratif. Ils sont implicitement créés lors de la définition des objets, c'est-à-dire des ensembles de méta-données correspondants à des types de données particuliers.

Comme illustré sur la figure 2a, une fonction 210 du gestionnaire de données 110 permet d'accéder aux liens et aux objets pour former une vue des

objets consistants, c'est-à-dire une vue présentant les relations entre des objets, qui peut être transmise à un utilisateur sous forme de vues de modèles de données 215.

De même, l'utilisateur peut visualiser des objets et des relations sous forme de vues d'objets 220 et de vues de relations 230, par exemple des listes.

Ainsi, selon ce mode de réalisation, l'utilisateur, c'est-à-dire la partie client du gestionnaire de données 110, dispose des trois types de vues suivants,

- une vue à travers les méta-données sur les objets contenus dans les différentes bases de données ;
- une vue sur les modèles de données créés dans l'architecture 100 ; et,
- une vue permettant de naviguer entre les différents objets à travers des liens définis dans la base de données des méta-données, ou qui leurs sont associés, tels que les liens de version, les liens utilisateur, les liens de configuration et les liens de production permettant la traçabilité.

Une description de type XML des objets de type *object proxy*, contenant toutes les méta-données associées à chaque donnée, est avantageusement créée pour être utilisée, par exemple, par le moteur d'exécution 115 afin d'accéder aux données mémorisées dans les zones de stockage 130 au cours du l'exécution de processus.

La figure 2b illustre un exemple d'algorithme permettant d'établir un arbre de données à partir des objets manipulés dans l'atelier orienté service collaboratif, ces objets représentant indifféremment des données réelles ou des processus.

Après avoir sélectionné un premier objet (étape 230), par exemple à travers une interface homme-machine, un lien vers un second objet, contenu dans ce premier objet, est accédé (étape 235). Ensuite, un lien vers un troisième objet, contenu dans ce second objet, est accédé (étape 240) pour permettre d'établir un lien entre le premier objet et le troisième (étape 245) ou entre les données associées au premier objet et celles associées au troisième objet.

Alternativement, un lien vers un troisième objet, contenu dans le premier objet, est accédé (étape 250) pour permettre d'établir un lien entre le second objet et le troisième (étape 255) ou entre les données associées au second objet et celles associées au troisième objet.

Comme suggéré par la flèche en trait pointillé, ces étapes peuvent être répétées, de façon itérative, pour déterminer un arbre de données selon lequel les feuilles représentent les données (ou les objets associés aux données) et les branches représentent les liens entre les données (ou entre les objets associés aux données).

La figure 3 représente schématiquement les éléments fonctionnels du moteur d'exécution 115, illustré sur la figure 1, ainsi que les informations transmises entre ces éléments, pour l'exécution d'un processus basé sur une représentation de type Python. Le moteur d'exécution 115 est activé par un objet consistant en un modèle de processus instancié 300 représenté ici sous forme de document de type XML.

Le modèle de processus instancié 300 est analysé dans un analyseur syntaxique 305 de type XML, appelé *XML parser* en terminologie anglo-saxonne, pour en extraire les informations permettant l'exécution du processus. En particulier, les paramètres, le code et les informations concernant les données nécessaires à l'exécution du processus sont extraits du document XML. De même, les données utilisées par le processus, formant les entrées du processus, sont retrouvées à l'aide de liens sur les objets concernés. Toutes les informations nécessaires à l'exécution du processus extraites du document XML sont transmises au noyau 310 du moteur d'exécution.

Le code du modèle de processus instancié 300, c'est-à-dire les données associées à l'objet représentant le modèle de processus instancié, correspond à des instructions directement exécutables par le moteur d'exécution, à des instructions exécutables à travers une interface spécifique, par exemple du code de type Python, ou à des fonctions ou des modules externes appelés par le moteur d'exécution tels que des fonctions ModelCenter.

L'exécution du processus associé au modèle de processus instancié 300 peut entraîner la création de nouveaux objets 315, formant la sortie du processus. Ces objets sont traités de la même façon que les objets décrits précédemment. En particulier, des méta-données et des liens mémorisés dans la base de données centralisée 125 ainsi que des données stockées dans des zones de stockage 130 sont associés à ces objets. Alternativement ou de façon complémentaire, l'entrée du processus peut être modifiée au cours de l'exécution de celui-ci. Les entrées modifiées formant une sortie du processus sont appelées des entrées/sorties.

Le noyau 310 du moteur d'exécution fait de préférence appel à un environnement 320 d'exécution, aussi appelé contexte d'exécution du modèle de processus instancié. L'environnement d'exécution a notamment pour objet de conserver un historique de l'exécution du processus (325), de rendre les données accessibles dans le format du moteur d'exécution (330), par exemple sous forme d'objets de type Python, de surveiller l'exécution du processus (335) et de fournir les interfaces 340 nécessaires à l'exécution de modules externes au noyau du moteur d'exécution.

Lorsqu'un processus est exécuté et, en particulier, lorsque de nouveaux objets sont créés, un ou plusieurs liens sont automatiquement créés dans les méta-données associées aux données utilisées par le processus (entrées, sorties et entrées/sorties) ainsi que dans les méta-données associées au modèle de processus instancié.

Avant qu'un processus ne soit exécuté, les méta-données associées au modèle de processus instancié comprennent des références vers les objets associés aux données devant être utilisées mais ne contiennent pas de lien. Les liens sont ainsi créés, dans chaque objet, lorsque le processus est exécuté.

Par exemple, lorsqu'un processus utilise une donnée pour produire un résultat, les liens suivants sont créés au cours de l'exécution du processus,

- un lien vers l'objet associé au modèle de processus est créé dans l'objet associé à la donnée d'entrée ;
- un lien vers l'objet associé à la donnée d'entrée est créé dans l'objet associé au modèle de processus ;

- un lien vers l'objet associé à la donnée de sortie est créé dans l'objet associé au modèle de processus ; et,

- un lien vers l'objet associé au modèle de processus est créé dans l'objet associé à la donnée de sortie.

La figure 4 illustre un exemple de mise en œuvre de l'atelier orienté service collaboratif conforme à l'invention, d'un point de vue applicatif, permettant l'exécution de code exécutable 400 issu de développements externes (405) et de développements spécifiques (410). Comme représenté, l'exécution de modèles de processus instanciés est récursive.

Selon la nature du code exécutable et l'implémentation de l'atelier orienté service collaboratif, le code exécutable est exécuté par l'atelier orienté service collaboratif ou par un applicatif externe. A titre d'illustration, le code exécutable est ici exécuté par l'atelier orienté service collaboratif s'il est de type Python et par un applicatif externe s'il est, par exemple, de type ModelCenter (MC).

L'exécution de fonctions externes à l'atelier orienté service collaboratif est réalisée selon un mécanisme d'encapsulation des fonctions externes dans des modèles de processus. La figure 5 illustre un exemple d'encapsulation d'un modèle ModelCenter dans un modèle de processus.

Si le code exécutable est de type Python, les données nécessaires à l'exécution du code, comprenant en particulier les références aux données d'entrée et de sortie ainsi qu'à l'environnement d'exécution, sont mises en forme selon le format Python (module 415). Le processus ainsi formaté est ensuite encapsulé (module 420) dans un modèle de processus pouvant être manipulé par l'atelier orienté service collaboratif comme un objet standard (module 425). Comme illustré par la flèche reliant le module 425 au module 415, le modèle de processus utilise l'interface Python lorsque le processus doit être exécuté.

De même, si le code exécutable est de type ModelCenter, les données nécessaires à l'exécution du code, comprenant en particulier les références aux données d'entrée et de sortie ainsi qu'à l'environnement d'exécution, sont encapsulées selon le format ModelCenter (module 430). Le

processus ainsi formaté est ensuite encapsulé (module 435) dans un modèle de processus pouvant être manipulé par l'atelier orienté service collaboratif comme un objet standard (module 425). A nouveau, comme illustré par la flèche reliant le module 425 au module 430, le modèle de processus utilise l'interface adaptée, ici l'interface ModelCenter, lorsque le processus doit être exécuté afin de permettre l'exécution du processus à l'extérieur de l'atelier orienté service collaboratif.

Comme suggéré par la référence 460, d'autres formats peuvent être utilisés.

L'instanciation du modèle de processus peut être réalisée après l'encapsulation du processus à l'aide d'une interface homme-machine, c'est-à-dire ici entre les modules 420 et 425 et entre les modules 435 et 425.

L'encapsulation a notamment pour objet d'adapter les modèles de processus selon un format prédéterminé. En particulier, l'encapsulation permet de contrôler la validité des données, de déterminer les paramètres nécessaires et d'imposer des valeurs par défaut aux paramètres manquants et de copier et/ou de générer le code approprié. De façon optionnelle, l'encapsulation permet également de définir des directives d'exécution, c'est-à-dire de définir, par exemple, quelles sont les machines sur lesquelles doivent être exécutés les processus ou certaines parties des processus.

L'encapsulation de fonctions externes dans un modèle de processus est de préférence réalisée à l'aide des sections XML suivantes,

- entrées et sorties : liens vers les objets associés aux données gérées dans l'atelier collaboratif permettant leur utilisation dans les fonctions externes ;
- paramètres : paramètres utilisés par les fonctions externes ; et,
- code : appel à la fonction externe.

La figure 5 illustre un exemple d'encapsulation d'un modèle ModelCenter 500 dans un modèle de processus 505. Comme représenté, un lien est établi entre chaque section du modèle de processus et les champs correspondants du modèle ModelCenter. L'appel au code exécutable du modèle ModelCenter est ici effectué à travers un script Python 510.

Ainsi, un modèle de processus peut faire appel à une fonction externe à l'atelier orienté service collaboratif. Du fait de la structure des objets associés aux modèles de processus et en particulier des liens, une traçabilité des données utilisées en entrée ou en sortie de modèles de processus faisant appel à des fonctions externes est possible.

De façon similaire, il est également possible d'encapsuler des modèles de processus dans des fonctions externes, par exemple dans des modèles ModelCenter. Ainsi, des modèles de processus développés dans l'atelier orienté service collaboratif peuvent être directement utilisés par des applicatifs externes. Via ce mécanisme, il est possible d'utiliser et de lier des modèles de processus à d'autres fonctions externes tout en conservant, lors de l'exécution, l'utilisation de données gérées dans l'atelier orienté service collaboratif ainsi que la traçabilité associée.

L'exécution de modèles de processus à partir d'un applicatif externe est réalisée par le module d'exécution neutre de la couche d'intégration de l'atelier orienté service collaboratif, appelé SRUN, utilisé pour exécuter les modèles de processus instanciés.

Un modèle de processus peut ainsi encapsuler des fonctions externes et peut être encapsulé dans des fonctions externes.

Dans ce double mécanisme d'encapsulation, le degré de traçabilité des données est ainsi déterminé par les fonctions exécutées par l'atelier orienté service collaboratif et par des applicatifs externes.

A titre d'illustration, si un modèle de processus A de l'atelier orienté service collaboratif fait appel à des fonctions B, C et D, B et C étant des modèles de processus de l'atelier orienté service collaboratif et D étant un modèle ModelCenter qui fait lui-même appel à des fonctions E et F, E étant une fonction externe et F étant un modèle de processus encapsulé de l'atelier orienté service collaboratif, il est possible de suivre les liens entre B, C et D mais pas entre E et F (simplement entre l'entrée de E et la sortie de F), exécutés à l'extérieur de l'atelier orienté service collaboratif. Il est ainsi possible de déterminer le niveau de granularité souhaité pour la traçabilité des données.

Comme indiqué précédemment, les données associées aux objets caractérisant des modèles de processus instanciés peuvent être représentés sous forme de fichiers, par exemple sous forme de fichier de type XML. Les descriptions des modèles de processus comprennent avantageusement plusieurs sections, spécifiques à chaque type d'information.

Les modèles de processus peuvent être composés eux-mêmes de sous-modèles de processus qui peuvent être mémorisés dans des sections des modèles de processus. Les sous-modèles de processus sont ici des références à des modèles de processus accessibles dans l'atelier orienté service collaboratif. Ainsi, il est possible de créer des modèles de processus composites faisant référence à plusieurs modèles de processus unitaires ou eux-mêmes composites.

Les attributs d'un modèle de processus instancié sont par exemple les suivants,

- la version du schéma de type XML permettant d'extraire les informations du modèle de processus ;
- l'identifiant, le nom, la version et le type du modèle de processus ;
- l'identifiant du modèle de processus instancié ;
- l'identifiant d'exécution de type ModelCenter si le modèle de processus instancié fait référence à une fonction de type ModelCenter ;
- les chemins d'accès aux mécanismes permettant l'exécution du processus de façon sécurisé (appelés *sandboxes* en terminologie anglo-saxonne) ;
- l'état d'exécution du modèle de processus qui spécifie dans quel état d'exécution est le modèle de processus instancié, par exemple soumis, en cours d'exécution, exécuté avec succès ou exécuté avec erreur ;
- l'état du modèle de processus définissant son état interne, par exemple primaire, partiellement instancié, instancié, en cours d'exécution ou exécuté ; et,
- des commentaires.

La section relative aux entrées, aux sorties, aux entrées/sorties et aux paramètres d'un modèle de processus instancié comprend par exemple le

nom, le type et la catégorie. D'autres informations peuvent être associées aux entrées, aux sorties, aux entrées/sorties et aux paramètres d'un modèle de processus instancié telles que le type d'utilisateur concerné, une description et une aide. Une section est utilisée pour chaque entrée, sortie, entrée/sortie et paramètre.

La section relative au contexte d'exécution d'un modèle de processus instancié comprend une section de classification spécifiant les outils utilisés durant l'exécution du modèle de processus instancié. La section relative au contexte d'exécution d'un modèle de processus instancié comprend également une section de création de contexte permettant de mémoriser le contexte d'exécution du modèle de processus instancié ainsi qu'une section de contexte courant d'exécution spécifiant la configuration devant être utilisée pour exécuter le processus et permettant de mémoriser les choix effectuer par le contrôleur de grille de calcul. La section relative au contexte d'exécution d'un modèle de processus instancié peut aussi comprendre une section de serveur d'exécution pour spécifier la configuration de la grille de calcul choisie lors de l'instanciation du modèle de processus ainsi qu'une section de rapport d'exécution remplie à la fin de l'exécution du processus contenant par exemple un code d'erreur ou les informations métiers relatives à l'exécution du processus. Les attributs des sections de création de contexte et de contexte courant d'exécution sont par exemple les suivants,

- le nom et le type de la machine utilisée pour exécuter le processus ainsi que son type de système d'exploitation ; et,
- la date de création du contexte ou d'exécution du processus.

La section relative aux données d'un modèle de processus instancié est prévue pour mémoriser les données privées du modèle de processus, disponibles lors de l'exécution du processus. Cette section peut comprendre elle-même des sections telles que des sections spécifiques à des processus particuliers tels que des processus de type ModelCenter.

La section relative au code d'un modèle de processus instancié comprend le code du processus devant être exécuté. Les attributs de cette

section sont par exemple le langage utilisé tel que le langage Python, la version du code et, éventuellement, des commentaires.

Un modèle de processus instancié peut, par ailleurs, comprendre une description de la liste des serveurs disponibles et/ou des serveurs préférés pour exécuter le processus.

La figure 6 illustre de façon synthétique un exemple de schéma 600 de type XML pour l'analyse d'un modèle de processus mémorisé sous forme de description de type XML. Le schéma de type XML 600 décrit les différentes sections contenues dans un fichier représentant un modèle de processus ainsi que les liens vers les sections de type XML des parties de gestion de données et de gestion de processus.

Les liens représentés en trait gras continu représentent l'inclusion, les liens en trait continu représentent la notion d'import et les liens en trait pointillé représentent la notion d'import implicite.

Les extensions de type PT (*Process Template*) sont associées aux modèles de processus, c'est-à-dire au moteur d'exécution, les extensions de type DM (*Data Management*) sont associées aux données, c'est-à-dire au gestionnaire de données, et les extensions de type MC sont associées à ModelCenter, c'est-à-dire à un gestionnaire particulier de processus.

Comme illustré, un modèle de processus 605 comprend plusieurs sous-ensembles 610 à 640 correspondant ici aux sections des entrées, des sorties, des paramètres, des serveurs, du contexte d'exécution, des données et du code, respectivement. Les sections liées aux entrées, aux sorties et aux paramètres comprennent elles-mêmes des arguments 645.

Le modèle de processus comprend implicitement les informations relatives aux modèles spécifiques 650-1 à 650-n tels que les modèles EDMG (sigle d'Editeur De Modèle Généralisé), ModelCenter et Excel (Excel est une marque). Le modèle ModelCenter du modèle de processus reçoit des informations d'un modèle particulier 655.

On remarque ici que le modèle Excel utilise une référence à la fonction devant être utilisée, par exemple une URL (sigle de *Uniform Resource*

Locator en terminologie anglo-saxonne) tandis que le modèle ModelCenter est, du fait de l'encapsulation, inclus dans le modèle de processus.

Les données spécifiées dans les sections 610, 615 et 620 sont issues des objets 660 qui comprennent eux-mêmes des liens (665 et 670) vers d'autres objets.

Ainsi, les références 600 à 650 sont associées au modèle de processus générique tandis que l'ensemble des références correspond à un modèle de processus instancié. Il convient néanmoins de remarquer ici que les paramètres 520 peuvent également être modifiés manuellement par l'utilisateur.

La figure 7 illustre un exemple d'algorithme 700 pour l'exécution de processus unitaires de modèles de processus instanciés.

Le modèle de processus instancié 705 est tout d'abord analysé (étape 710) pour contrôler sa validité à l'aide d'un schéma XML prédéterminé 715. S'il n'est pas valide, l'exécution du processus est arrêtée et un autre modèle de processus instancié peut être traité. Si le modèle de processus instancié est valide, les informations contenues dans ce modèle sont extraites (étape 720).

L'extraction des informations permet notamment de déterminer la configuration du contexte d'exécution 725 ainsi que l'environnement d'exécution 730 pouvant notamment comprendre les entrées, les sorties, les entrées/sorties, les paramètres et/ou les données propres au modèle de processus instancié. Selon un mode de réalisation préféré, la configuration du contexte d'exécution 725 et l'environnement d'exécution sont déterminés selon le format Python.

La configuration du contexte d'exécution est ensuite contrôlée selon la validité du domaine d'exécution (étape 735). En effet, le créateur d'un processus a la possibilité de spécifier un domaine de validité, par exemple une version spécifique de logiciel. Si un utilisateur veut utiliser une autre version, il peut être alerté ou contraint à changer son choix, la condition de validité du domaine d'exécution n'étant pas satisfaite. Si la configuration n'est pas valide, l'exécution du processus est arrêtée et un autre modèle de processus instancié peut être traité.

Si la configuration est valide, les conditions sur les entrées et les entrées/sorties sont vérifiées (étape 740). Lors de l'instanciation du processus il se peut que l'utilisateur ait référencé un objet qui ne peut pas être utilisé dans ce processus car il ne satisfait pas les conditions définies par le créateur du processus. Dans l'exemple donné concernant la table et la chaise, le processus de déménagement peut, par exemple, avoir comme condition une couleur spécifique de la chaise. Si l'utilisateur choisit une chaise d'une autre couleur, le processus de déménagement ne peut pas avoir lieu. Si les conditions sur les entrées et les entrées/sorties ne sont pas valides, l'exécution du processus est arrêtée et un autre modèle de processus instancié peut être traité.

Si les conditions sur les entrées et les entrées/sorties sont valides, l'exécution du processus est mise en œuvre selon le code du modèle de processus instancié (étape 745), c'est-à-dire selon les données associées à l'objet représentant le modèle de processus instancié.

Le code du modèle de processus instancié est ensuite exécuté (étape 750). De façon générale, le noyau du module d'exécution, appelé SRUN, charge en mémoire les entrées, les entrées/sorties ainsi que les modules Python nécessaires à l'exécution du code. Si nécessaire, un environnement d'exécution distribué, basé sur les capacités du module d'exécution SRUN, peut être mis en œuvre.

Dans le cas des modèles de processus instanciés composites, le module d'exécution SRUN créé en mémoire des instances des sous modèles de processus et les exécute successivement comme des modèles de processus instanciés unitaires. La cohérence entre les différents processus unitaires est alors assurée par le moteur d'exécution du flux de travaux ou *workflow*.

Durant l'exécution du processus, le script contenant le code est exécuté dans l'environnement d'exécution correspondant, éventuellement sur une machine différente de celle du gestionnaire de données, en fonction du choix du gestionnaire de la grille de calcul. De préférence, les mécanismes permettant l'exécution du processus de façon sécurisée sont utilisés pour lire et

écrire les données physiques, pour mémoriser l'historique de l'exécution du processus et pour la mémorisation temporaire des données.

Après l'exécution du script, c'est-à-dire du processus, le moteur d'exécution transmet les sorties au domaine approprié et mémorise le rapport d'exécution.

La figure 8 illustre un exemple de traitement de données 800 pouvant être effectué dans un atelier orienté service collaboratif conforme à l'invention. Selon cet exemple, des données 805 correspondent à des informations relatives à une maison en ruine. Un objet, faisant référence à des méta-données, est associé à ces données, précisant l'état en ruine ainsi que d'autres informations jugées comme pertinentes, ici le nombre de pièces et le nombre de fenêtres. Des données 810 correspondent aux informations relatives à une maison habitable. A nouveau, un objet, faisant référence à des méta-données, est associé à ces données, précisant un état, ici l'état habitable, ainsi que des informations jugées comme pertinentes, c'est-à-dire ici le nombre de pièces et le nombre de fenêtres. Les données 810 peuvent être obtenues à partir de l'objet lié aux données 805, et donc à partir des données 805, à l'aide d'un objet représentant un modèle de processus instancié de rénovation 815.

De façon similaire, l'objet correspondant aux données 810 peut être combiné à un objet de type salle à manger lié aux données 820 et à un objet de type lit lié aux données 825 dans un modèle de processus instancié de déménagement 830 pour créer un objet de type maison habitée lié aux données 835. Les méta-données liées aux données 835 précisent ici l'état habité ainsi que le nombre de pièces et de fenêtres.

L'exemple donné en annexe sous la référence « extrait de code 3 » illustre un exemple simplifié de méta-données associées aux données 805 de la figure 8 avant l'exécution du processus de rénovation.

Comme indiqué dans cet exemple, les méta-données comprennent des informations génériques, notamment le type et le nom de l'objet ainsi que le lien pour accéder à la donnée correspondante, et des informations spécifiques, ici l'état, le nombre de fenêtres et le nombre de pièce. Il convient de noter que

les méta-données sont du type IN, c'est-à-dire qu'elles ne correspondent pas à des données créées par un modèle de processus instancié.

Le modèle de processus de rénovation pouvant être utilisé pour traiter l'exemple précédent représentant une maison en ruine peut être décrit génériquement tel que présenté en annexe sous la référence « extrait de code 4 ».

Ce modèle de processus prend en entrée un objet de type maison et créé en sortie un autre objet de type maison. Ce modèle de processus peut être instancié pour contenir, notamment, des références aux objets associés aux données 805 et 810 de la figure 8. Le modèle de processus instancié est alors, par exemple, celui donné en annexe sous la référence « extrait de code 5 ».

Comme indiqué, le modèle de processus instancié comprend en outre des paramètres d'exécution, l'identification de la machine d'exécution, la configuration applicative utilisée ainsi que les liens d'historiques d'exécution entre les objets associés aux données 805 et au modèle de processus instancié 815 et entre les objets associés au modèle de processus instancié 815 et aux données 810.

Après l'exécution du processus correspondant à ce modèle de processus instancié, les méta-données liées aux données 805 sont modifiées pour ajouter un lien d'historique permettant la traçabilité des données. Les méta-données associées aux données 805 peuvent donc, après l'exécution du processus, être représentées de la façon donnée en annexe sous la référence « extrait de code 6 ».

Comme précisé dans cet exemple, un lien a été créé entre les objets associés aux données 805 et au modèle de processus instancié 815. Ainsi, à l'aide de l'objet associé au modèle de processus instancié 815, il est possible d'établir un lien entre les objets associés aux données 805 et 810.

De même, après l'exécution du processus, les données 810 ainsi que les méta-données associées à celles-ci, incluant les liens, sont créées. Les méta-données associées aux données 810 peuvent alors être représentées de la façon donnée en annexe sous la référence « extrait de code 7 ».

Comme précédemment, un lien a été créé entre l'objet associé aux données 810 et l'objet associé au modèle de processus instancié 815. Ainsi, à l'aide de l'objet associé au modèle de processus instancié 815, il est possible d'établir le lien existant entre les données 805 et 810. Il convient de noter que les méta-données sont ici du type OUT, c'est-à-dire qu'elles correspondent à des données créées par un modèle de processus instancié dans l'atelier orienté service collaboratif.

De la même façon, un second modèle de processus peut être créé et instancié pour traiter les données 810, 820 et 825 pour créer les données 835 et l'objet associé. Lorsque le second modèle de processus instancié 830 caractérisant un processus de déménagement est exécuté, les méta-données associées aux données 810 sont modifiées pour intégrer le lien induit par l'exécution du modèle de processus instancié 830. Les méta-données associées aux données 810 peuvent alors s'exprimer sous la forme donnée en annexe sous la référence « extrait de code 8 ».

Le lien créé entre l'objet associé aux données 810 et l'objet associé au modèle de processus instancié 830 permet, à l'aide de cet objet, d'établir le lien entre les objets associés aux données 810 et 835.

Par ailleurs, lors de l'exécution du second modèle de processus instancié 830, les données 835 sont créées ainsi que l'objet associé. Les méta-données associées peuvent être représentées sous la forme donnée en annexe sous la référence « extrait de code 9 ».

Le lien créé entre l'objet associé aux données 835 et l'objet associé au modèle de processus instancié 830 permet, à l'aide de cet objet, de déterminer les liens entre les objets associés aux données 835 et 810, entre les objets associés aux données 835 et 820 ainsi qu'entre les objets associés aux données 835 et 825.

Un dispositif adapté à mettre en œuvre une partie de l'invention est illustré sur la figure 9. Le dispositif 900 est par exemple un micro-ordinateur, un ordinateur ou une station de travail.

Le dispositif 900 comporte ici un bus de communication 902 auquel sont reliés :

- une unité centrale de traitement ou microprocesseur 903 (CPU, sigle de *Central Processing Unit* en terminologie anglo-saxonne) ;

- une mémoire morte 904 (ROM, acronyme de *Read Only Memory* en terminologie anglo-saxonne) pouvant comporter les programmes "Prog", "Prog1" et "Prog2" ;

- une mémoire vive ou mémoire cache 906 (RAM, acronyme de *Random Access Memory* en terminologie anglo-saxonne) comportant des registres adaptés à enregistrer des variables et paramètres créés et modifiés au cours de l'exécution des programmes précités ; et,

- une interface de communication 918 adaptée à transmettre et à recevoir des données.

Optionnellement, le dispositif 900 peut également disposer :

- d'un écran 908 permettant de visualiser des données et/ou de servir d'interface graphique avec l'utilisateur qui pourra interagir avec les programmes selon l'invention, à l'aide d'un clavier et d'une souris 910 ou d'un autre dispositif de pointage, un écran tactile ou une télécommande ;

- d'un disque dur 912 pouvant comporter les programmes "Prog", "Prog1" et "Prog2" précités et des données traitées ou à traiter selon l'invention ; et,

- d'un lecteur de cartes mémoires 914 adapté à recevoir une carte mémoire 916 et à y lire ou à y écrire des données traitées ou à traiter selon l'invention.

Le bus de communication permet la communication et l'interopérabilité entre les différents éléments inclus dans le dispositif 900 ou reliés à lui. La représentation du bus n'est pas limitative et, notamment, l'unité centrale est susceptible de communiquer des instructions à tout élément du dispositif 900 directement ou par l'intermédiaire d'un autre élément du dispositif 900.

Le code exécutable de chaque programme permettant au dispositif programmable de mettre en œuvre les processus selon l'invention, peut être stocké, par exemple, dans le disque dur 912 ou en mémoire morte 904.

Selon une variante, la carte mémoire 916 peut contenir des données ainsi que le code exécutable des programmes précités qui, une fois lu par le dispositif 900, sera stocké dans le disque dur 912.

Selon une autre variante, le code exécutable des programmes pourra être reçu, au moins partiellement, par l'intermédiaire de l'interface 918, pour être stocké de façon identique à celle décrite précédemment.

De manière plus générale, le ou les programmes pourront être chargés dans un des moyens de stockage du dispositif 900 avant d'être exécutés.

L'unité centrale 903 va commander et diriger l'exécution des instructions ou portions de code logiciel du ou des programmes selon l'invention, instructions qui sont stockées dans le disque dur 912 ou dans la mémoire morte 904 ou bien dans les autres éléments de stockage précités. Lors de la mise sous tension, le ou les programmes qui sont stockés dans une mémoire non volatile, par exemple le disque dur 912 ou la mémoire morte 904, sont transférés dans la mémoire vive 906 qui contient alors le code exécutable d'au moins une partie du ou des programmes selon l'invention, ainsi que des registres pour mémoriser les variables et paramètres nécessaires à la mise en œuvre de l'invention.

Naturellement, pour satisfaire des besoins spécifiques, une personne compétente dans le domaine de l'invention pourra appliquer des modifications dans la description précédente.

ANNEXE

Extrait de code 1 :

```
<Atelier.DM:FrameworkMetaData>
    <Atelier.DM:Attributes.Generic />
    <Atelier.DM:Attributes.Specific />
</Atelier.DM:FrameworkMetaData>
```

Extrait de code 2 :

```
<Atelier.DM:FrameworkMetaData
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM">
    <Atelier.DM:Attributes.Generic acl_name="Prive" comments="None"
creation_date="2006-04-27 10:29:43" db_type="ELEMENT"
external_id="chaise_numero_0010100432325" id="1411" lock_owner="None" lock_state="0"
mdb_instance="MDB_exploit" modification_date="2006-04-27 10:29:43" name="chaise1"
owner="nom_prenom" publication_level="Prive" shared_state="None" site="Toulouse"
source="LOCAL" status="Prive" version_comments="None" version_number="None"
version_owner="None" version_state="None" version_tag="None"/>
    <Atelier.DM:Attributes.Specific nb_pieds="4" epaisseur="5cm"
couleur="rouge" matiere="bois" poids="2kg"/>
</Atelier.DM:FrameworkMetaData>
```

Extrait de code 3 :

```
<Atelier.DM:EEObjectProxy xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM"
publish="0" state="0" type="IN">
  <Atelier.DM:OriginalPointer>
    <Atelier.DM:Pointer external_id="description pour accès à la donnée physique"/>
  </Atelier.DM:OriginalPointer>
  <Atelier.DM:ObjectProxy type="MAISON">
    <Atelier.DM:Pointer external_id="description pour accès à la donnée physique"/>
    <Atelier.DM:FrameworkMetaData
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM">
      <Atelier.DM:Attributes.Generic acl_name="Prive" db_type="MAISON"
external_id="description pour accès à la donnée physique" id="identifiant"
mdb_instance="MDB_exploit" modification_date="" name="ma maison bleue" source="CMSDK"
status="Private"/>
      <Atelier.DM:Attributes.Specific etat="en ruine" nb_fenetre="4" nb_pieces="10"/>
    </Atelier.DM:FrameworkMetaData>
  </Atelier.DM:ObjectProxy>
</Atelier.DM:EEObjectProxy>
```

Extrait de code 4 :

```
<?xml version="1.0" ?>
  <Atelier.PT:ProcessTemplate comment="mon premier traitement" creator="createur"
id="identifiant" name="Renovation maison" schemaVersion="0.1" state="primary"
execution_state="not_submitted" type="action" version="0.1"
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM"
xmlns:Atelier.MC="http://www.societe.com/egat/Atelier/MC"
xmlns:Atelier.PT="http://www.societe.com/egat/Atelier/PT"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://www.societe.com/egat/Atelier/PT Atelier.PT.ProcessTemplate.xsd
http://www.societe.com/egat/Atelier/DM Atelier.DM.xsd">
    <Atelier.PT:SandboxPointer external_id="" source_name="" source_site=""
source_type="" type=""/>
    <Atelier.PT:Input category="object" desc="maison en ruine" name="" type="MAISON">
      <Atelier.PT:Condition>
        <![CDATA[nombre de fenetres > 3]]>
      </Atelier.PT:Condition>
    </Atelier.PT:Input>
    <Atelier.PT:Output category="object" desc="maison habitable" name="" type="MAISON">
    </Atelier.PT:Output>
    <Atelier.PT:Parameter desc="paramètres d exécution" name="para_list" type="list"/>
    <Atelier.PT:Server>
      <Atelier.PT:ValidServer>
        <![CDATA[mot clé et valeurs décrivant les caractéristiques machines]]>
      </Atelier.PT:ValidServer>
      <Atelier.PT:DefaultServer>
        <![CDATA[mot clé et valeurs caractéristiques de la machine par défaut]]>
      </Atelier.PT:DefaultServer>
    </Atelier.PT:Server>
    <Atelier.PT:ExecutionContext>
      <Atelier.PT:Classification processType="batch">
        <Atelier.PT:Suite name="Suite1"/>
        <Atelier.PT:Tool name="Tool1"/>
      </Atelier.PT:Classification>
      <Atelier.PT:CreationContext class="" date="" hostname="" osname="" type="">
        <![CDATA[[configuration] mot clé et valeur décrivant la configuration applicative par
defaut
        [validity] condition de validite de la configuration applicative]]>
      </Atelier.PT:CreationContext>
    </Atelier.PT:ExecutionContext>
  <Atelier.PT:Data/>
  <Atelier.PT:Code comment="commentaire sur le code" langage="python" version="">
    <![CDATA[Code a executer]]>
  </Atelier.PT:Code>
        </Atelier.PT:ProcessTemplate>
```

Extrait de code 5 :

```
<?xml version="1.0" ?>
  <Atelier.PT:ProcessTemplate comment="mon premier traitement" creator="createur"
id="identifiant" name="Renovation maison" schemaVersion="0.1" state="primary"
execution_state="not_submitted" type="action" version="0.1"
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM"
xmlns:Atelier.MC="http://www.societe.com/egat/Atelier/MC"
xmlns:Atelier.PT="http://www.societe.com/egat/Atelier/PT"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.societe.com/egat/Atelier/PT Atelier.PT.ProcessTemplate.xsd
http://www.societe.com/egat/Atelier/DM Atelier.DM.xsd">
    <Atelier.PT:SandboxPointer external_id="" source_name=""/>
    <Atelier.PT:Input category="object" desc="maison en ruine" name="Donnée_805"
type="MAISON">
      <Atelier.PT:Condition>
        <![CDATA[ nombre de fenetres > 3 ]]>
      </Atelier.PT:Condition>
      <Atelier.DM:EEObjectProxy publish="0" state="0" type="IN">
        <Atelier.DM:OriginalPointer>
```

```
                <Atelier.DM:Pointer external_id="description pour accès à la donnée
physique"/>
            </Atelier.DM:OriginalPointer>
            <Atelier.DM:ObjectProxy type="MAISON">
                <Atelier.DM:Pointer external_id="description pour accès à la donnée
physique"/>
                <Atelier.DM:FrameworkMetaData
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM">
                    <Atelier.DM:Attributes.Generic acl_name="Prive" db_type="MAISON"
external_id="description pour accès à la donnée physique" id="identifiant"
mdb_instance="MDB_exploit" modification_date="" name="ma maison bleue" source="CMSDK"
status="Private"/>
                    <Atelier.DM:Attributes.Specific etat="en ruine" nb_fenetre="4"
nb_pieces="10" />
                    <Atelier.DM:Links>
                        <Atelier.DM:Link comments="description du lien" date=""
id_FO_1="identifiant donnée 805" id_FO_2="identifiant IPT 815" mdb_instance="MDB_exploit"
owner="to39751" subtype="sous type de lien" type="HISTORY"
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" />
                    </Atelier.DM:Links>
                </Atelier.DM:FrameworkMetaData>
            </Atelier.DM:ObjectProxy>
        </Atelier.DM:EEObjectProxy>
    </Atelier.PT:Input>
    <Atelier.PT:Output category="object" desc="maison habitable" name="Donnée_810"
type="MAISON">
        <Atelier.DM:EEObjectProxy publish="1" state="1" type="OUT">
            <Atelier.DM:SandboxPointer>
                <Atelier.DM:Pointer external_id="description pour accéder au bac a sable"
source_name="" source_site="" source_type="" type="" />
            </Atelier.DM:SandboxPointer>
            <Atelier.DM:FinalPointer>
                <Atelier.DM:Pointer external_id="description pour accès à la donnée
physique"/>
            </Atelier.DM:FinalPointer>
            <Atelier.DM:ObjectProxy type="MAISON">
                <Atelier.DM:Pointer external_id="description pour accès à la donnée
physique"/>
                <Atelier.DM:FrameworkMetaData
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM">
                    <Atelier.DM:Attributes.Generic acl_name="Prive" db_type="MAISON"
external_id="description pour accès à la donnée physique" id="identifiant"
mdb_instance="MDB_exploit" modification_date="" name="ma maison bleue" source="CMSDK"
status="Private"/>
                    <Atelier.DM:Attributes.Specific etat="habitable" nb_fenetre="6"
nb_pieces="12" />
                    <Atelier.DM:Links>
                        <Atelier.DM:Link comments="description du lien" date=""
id_FO_1="identifiant IPT 815" id_FO_2="identifiant Donnée 810" mdb_instance="MDB_exploit"
owner="to39751" subtype="sous type de lien" type="HISTORY"
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" />
                    </Atelier.DM:Links>
                </Atelier.DM:FrameworkMetaData>
            </Atelier.DM:ObjectProxy>
        </Atelier.DM:EEObjectProxy>
    </Atelier.PT:Output>
    <Atelier.PT:Parameter desc="parametres d execution" name="para_list" type="list">
        <Atelier.PT:ParameterValue>
```

```
            <![CDATA[ parametre d execution ]]>
        </Atelier.PT:ParameterValue>
    </Atelier.PT:Parameter>
    <Atelier.PT:Server>
        <Atelier.PT:ValidServer>
            <![CDATA[ mot clé et valeurs décrivant les caractéristiques machines ]]>
        </Atelier.PT:ValidServer>
        <Atelier.PT:DefaultServer>
            <![CDATA[ mot clé et valeurs caractéristiques de la machine par défaut ]]>
        </Atelier.PT:DefaultServer>
    </Atelier.PT:Server>
    <Atelier.PT:ExecutionContext>
        <Atelier.PT:Classification processType="batch">
            <Atelier.PT:Suite name="Suite1" />
            <Atelier.PT:Tool name="Tool1" />
        </Atelier.PT:Classification>
        <Atelier.PT:CreationContext class="" date="" hostname="" osname="" type="">
            <![CDATA[[configuration] mot clé et valeur décrivant la configuration applicative par
défaut
                [validity] condition de validité de la configuration applicative ]]>
        </Atelier.PT:CreationContext>
        <Atelier.PT:RunContext class="" date="" hostname="" osname="" type="">
            <![CDATA[[configuration] mot clé et valeur décrivant la configuration applicative
utilisée]]>
        </Atelier.PT:RunContext>
        <Atelier.PT:RunServer>
            <![CDATA[ mot clé et valeur décrivant la machine d'exécution utilisée ]]>
        </Atelier.PT:RunServer>
        <Atelier.PT:ExecutionReport return_code="0">
            <![CDATA[ Rapport d execution ]]>
        </Atelier.PT:ExecutionReport>
    </Atelier.PT:ExecutionContext>
    <Atelier.PT:Data />
    <Atelier.PT:Code comment="commentaire sur le code" langage="python" version="">
        <![CDATA[ Code a exécuter ]]>
    </Atelier.PT:Code>
            </Atelier.PT:ProcessTemplate>
```

Extrait de code 6 :

```
<Atelier.DM:EEObjectProxy xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM"
publish="0" state="0" type="IN">
    <Atelier.DM:OriginalPointer>
        <Atelier.DM:Pointer external_id="description pour accès à la donnée physique"
source_name="" source_site="" source_type="" type="" />
    </Atelier.DM:OriginalPointer>
    <Atelier.DM:ObjectProxy type="MAISON">
        <Atelier.DM:Pointer external_id="description pour accès à la donnée physique"/>
        <Atelier.DM:FrameworkMetaData
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM">
            <Atelier.DM:Attributes.Generic acl_name="Prive" db_type="MAISON"
external_id="description pour accès à la donnée physique" id="identifiant"
mdb_instance="MDB_exploit" modification_date="" name="ma maison bleue" source="CMSDK"
status="Private"/>
            <Atelier.DM:Attributes.Specific etat="en ruine" nb_fenetre="4" nb_pieces="10" />
            <Atelier.DM:Links>
```

<Atelier.DM:Link comments="description du lien" date="" id_FO_1="identifiant Donnée 805" id_FO_2="identifiant IPT 815" mdb_instance="MDB_exploit" owner="to39751" subtype="sous type de lien" type="HISTORY" xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" />
</Atelier.DM:Links>
</Atelier.DM:FrameworkMetaData>
</Atelier.DM:ObjectProxy>
</Atelier.DM:EEObjectProxy>

Extrait de code 7 :

<Atelier.DM:EEObjectProxy xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" publish="1" state="1" type="OUT">
<Atelier.DM:SandboxPointer>
<Atelier.DM:Pointer external_id="description pour accès au bac a sable"/>
</Atelier.DM:SandboxPointer>
<Atelier.DM:FinalPointer>
<Atelier.DM:Pointer external_id="description pour accès à la donnée physique"/>
</Atelier.DM:FinalPointer>
<Atelier.DM:ObjectProxy type="MAISON">
<Atelier.DM:Pointer external_id="description pour accès à la donnée physique"/>
<Atelier.DM:FrameworkMetaData xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM">
<Atelier.DM:Attributes.Generic acl_name="Prive" db_type="MAISON" external_id="description pour accès à la donnée physique" id="identifiant" mdb_instance="MDB_exploit" modification_date="" name="ma maison bleue" source="CMSDK" status="Private"/>
<Atelier.DM:Attributes.Specific etat="habitable" nb_fenetre="6" nb_pieces="12" />
<Atelier.DM:Links>
<Atelier.DM:Link comments="description du lien" date="" id_FO_1="identifiant IPT 815" id_FO_2="identifiant Donnée 810" mdb_instance="MDB_exploit" owner="to39751" subtype="sous type de lien" type="HISTORY" xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" />
</Atelier.DM:Links>
</Atelier.DM:FrameworkMetaData>
</Atelier.DM:ObjectProxy>
</Atelier.DM:EEObjectProxy>

Extrait de code 8 :

<Atelier.DM:EEObjectProxy xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" publish="1" state="1" type="OUT">
<Atelier.DM:SandboxPointer>
<Atelier.DM:Pointer external_id="description pour accès au bac à sable" source_name="" source_site="" source_type="" type="" />
</Atelier.DM:SandboxPointer>
<Atelier.DM:FinalPointer>
<Atelier.DM:Pointer external_id="description pour accès à la donnée physique"/>
</Atelier.DM:FinalPointer>
<Atelier.DM:ObjectProxy type="MAISON">
<Atelier.DM:Pointer external_id="description pour accès à la donnée physique"/>
<Atelier.DM:FrameworkMetaData xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM">
<Atelier.DM:Attributes.Generic acl_name="Prive" db_type="MAISON" external_id="description pour accès à la donnée physique" id="identifiant"

```
mdb_instance="MDB_exploit" modification_date="" name="ma maison bleue" source="CMSDK"
status="Private"/>
            <Atelier.DM:Attributes.Specific etat="habitable" nb_fenetre="6" nb_pieces="12" />
            <Atelier.DM:Links>
                <Atelier.DM:Link comments="description du lien" date="" id_FO_1="identifiant IPT
815" id_FO_2="identifiant Donnée 810" mdb_instance="MDB_exploit" owner="to39751"
subtype="sous type de lien" type="HISTORY"
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" />
                <Atelier.DM:Link comments="description du lien" date="" id_FO_1="identifiant
Donnée 810" id_FO_2="identifiant IPT 830" mdb_instance="MDB_exploit" owner="to39751"
subtype="sous type de lien" type="HISTORY"
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" />
            </Atelier.DM:Links>
        </Atelier.DM:FrameworkMetaData>
    </Atelier.DM:ObjectProxy>
                </Atelier.DM:EEObjectProxy>
```


Extrait de code 9 :


```
<Atelier.DM:EEObjectProxy xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM"
publish="1" state="1" type="OUT">
    <Atelier.DM:SandboxPointer>
        <Atelier.DM:Pointer external_id="description pour accès au bac à sable"/>
    </Atelier.DM:SandboxPointer>
    <Atelier.DM:FinalPointer>
        <Atelier.DM:Pointer external_id="description pour accès à la donnée physique"/>
    </Atelier.DM:FinalPointer>
    <Atelier.DM:ObjectProxy type="MAISON">
        <Atelier.DM:Pointer external_id="description pour accès à la donnée physique"/>
        <Atelier.DM:FrameworkMetaData
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM">
            <Atelier.DM:Attributes.Generic acl_name="Prive" db_type="MAISON"
external_id="description pour accès à la donnée physique" id="identifiant"
mdb_instance="MDB_exploit" modification_date="" name="ma maison bleue" source="CMSDK"
status="Private"/>
            <Atelier.DM:Attributes.Specific etat="habite" nb_fenetre="6" nb_pieces="12" />
            <Atelier.DM:Links>
                <Atelier.DM:Link comments="description du lien" date="" id_FO_1="identifiant IPT
830" id_FO_2="identifiant Donnée 835" mdb_instance="MDB_exploit" owner="to39751"
subtype="sous type de lien" type="HISTORY"
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" />
                <Atelier.DM:Link comments="description du lien" date="" id_FO_1="identifiant IPT
830" id_FO_2="identifiant Donnée 835" mdb_instance="MDB_exploit" owner="to39751"
subtype="sous type de lien" type="HISTORY"
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" />
                <Atelier.DM:Link comments="description du lien" date="" id_FO_1="identifiant IPT
830" id_FO_2="identifiant Donnée 835" mdb_instance="MDB_exploit" owner="to39751"
subtype="sous type de lien" type="HISTORY"
xmlns:Atelier.DM="http://www.societe.com/egat/Atelier/DM" />
            </Atelier.DM:Links>
        </Atelier.DM:FrameworkMetaData>
    </Atelier.DM:ObjectProxy>
                </Atelier.DM:EEObjectProxy>
```

Table 1 :

méta-données							
génériques				spécifiques			
ID	date	URL	type	nb pieds	mat.	coul.	poids
1	12/09/07	http://x.y/z/	élément	3	bois	rouge	3500
...	...	...	...	...	...	...	...
n	29/01/08	http://m.n/o/p/q	élément	4	métal	vert	1800

Table 2 :

ID1	5	8	3	5	...	...	...	...
ID2	8	5	1	1	...	...	...	...

REVENDICATIONS

1. Procédé de gestion de données dans un atelier orienté service collaboratif mis en œuvre dans un système informatique, ce procédé étant caractérisé en ce que ledit atelier orienté service collaboratif est adapté à traiter des objets associés à des données représentatives de données réelles ou de processus et en ce que le procédé comprend les étapes suivantes,

- sélection (230) d'au moins un premier objet comprenant au moins un lien, appelé premier lien, vers un second objet, distinct dudit premier objet ;

- accès (240, 245) à au moins un lien vers un troisième objet, appelé second lien, ledit second lien étant compris dans l'un desdits premier et second objets, ledit troisième objet étant distinct desdits premier et second objets ; et,

- établissement (245, 255) d'un lien entre les données associées audit troisième objet et l'autre desdits premier et deuxième objets.

2. Procédé selon la revendication 1 selon lequel les données associées audit premier objet sont représentatives d'un processus.

3. Procédé selon la revendication 2 selon lequel les données associées audit second objet sont au moins partiellement obtenues à partir dudit processus ou selon lequel ledit processus est appliqué sur au moins une partie desdites données associées audit second objet.

4. Procédé selon l'une quelconque des revendications précédentes comprenant en outre une étape de construction (215) d'un arbre de données selon lesdits premier et second liens.

5. Procédé selon l'une quelconque des revendications précédentes caractérisé en ce que lesdites étapes d'accès à au moins un lien et d'établissement d'un lien sont répétées de façon récursive pour établir un lien entre des données associées à au moins deux objets distincts.

6. Procédé selon l'une quelconque des revendications précédentes selon lequel au moins l'un desdits liens contenu dans l'un desdits objets est mémorisé dans une table de liens.

7. Procédé selon l'une quelconque des revendications précédentes selon lequel ledit processus fait appel à au moins une fonction extérieure audit atelier orienté service collaboratif.

8. Procédé selon l'une quelconque des revendications précédentes selon lequel lesdits objets sont mémorisés dans une base de données centralisée (125).

9. Procédé selon l'une quelconque des revendications précédentes selon lequel lesdites données associées auxdits objets sont mémorisées dans des dispositifs distants (130).

10. Programme d'ordinateur comprenant des instructions adaptées à la mise en œuvre de chacune des étapes du procédé selon l'une quelconque des revendications précédentes.

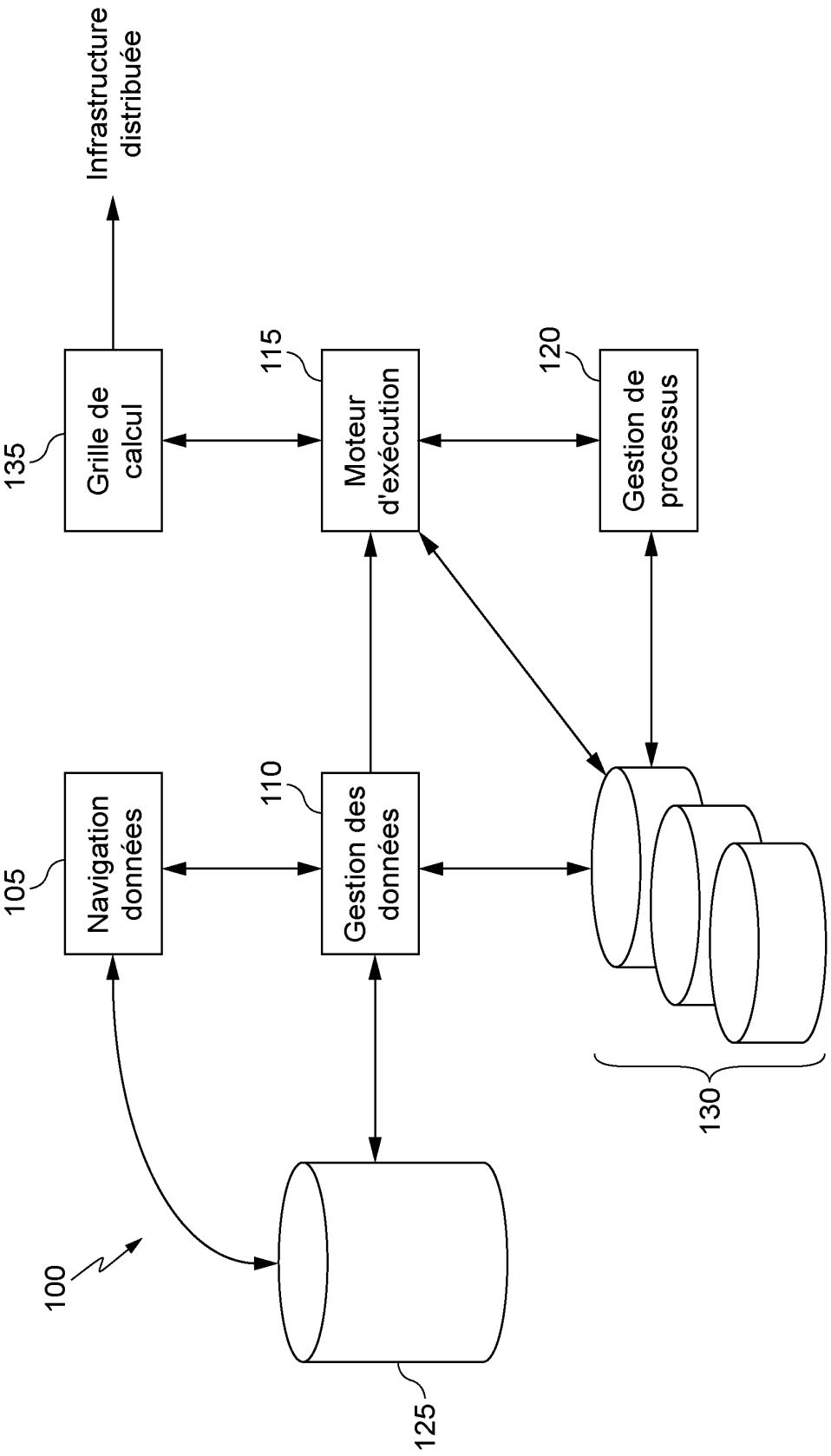

Infrastructure distribuée
Grille de calcul
135
Moteur d'exécution
115
Gestion de processus
120
Navigation données
105
Gestion des données
110
130
100
125

Fig. 1

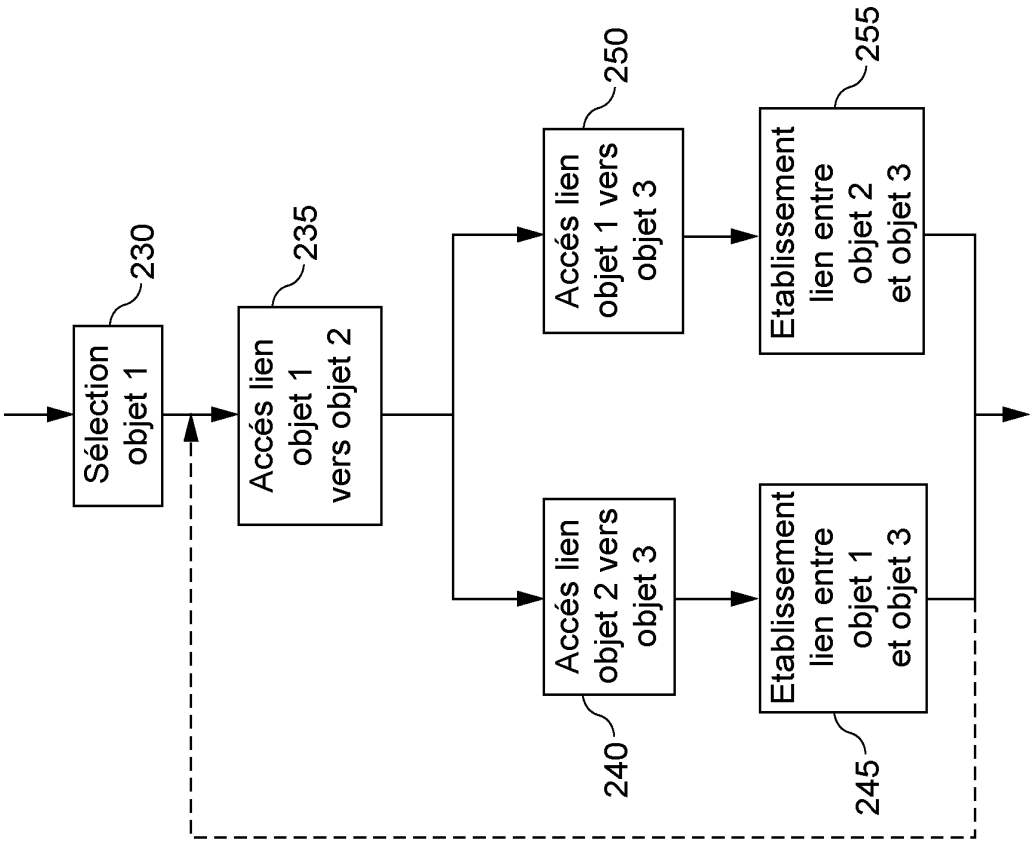

Sélection objet 1
230
Accès lien objet 1 vers objet 2
235
Accès lien objet 1 vers objet 3
250
Etablissement lien entre objet 2 et objet 3
255
Accès lien objet 2 vers objet 3
240
Etablissement lien entre objet 1 et objet 3
245

Fig. 2b

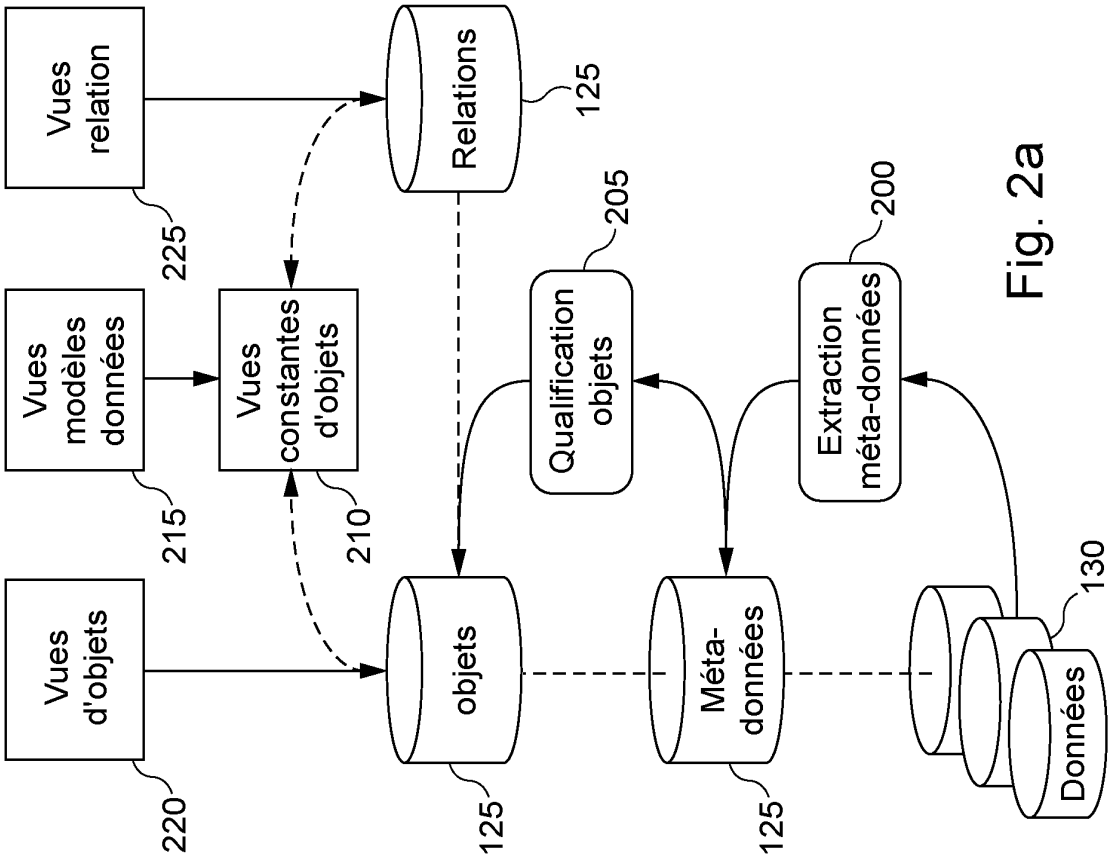

Vues relation
225
Relations
125
Vues modèles données
215
Vues constantes d'objets
210
Qualification objets
205
Extraction méta-données
200
Vues d'objets
220
objets
125
Méta-données
125
Données
130

Fig. 2a

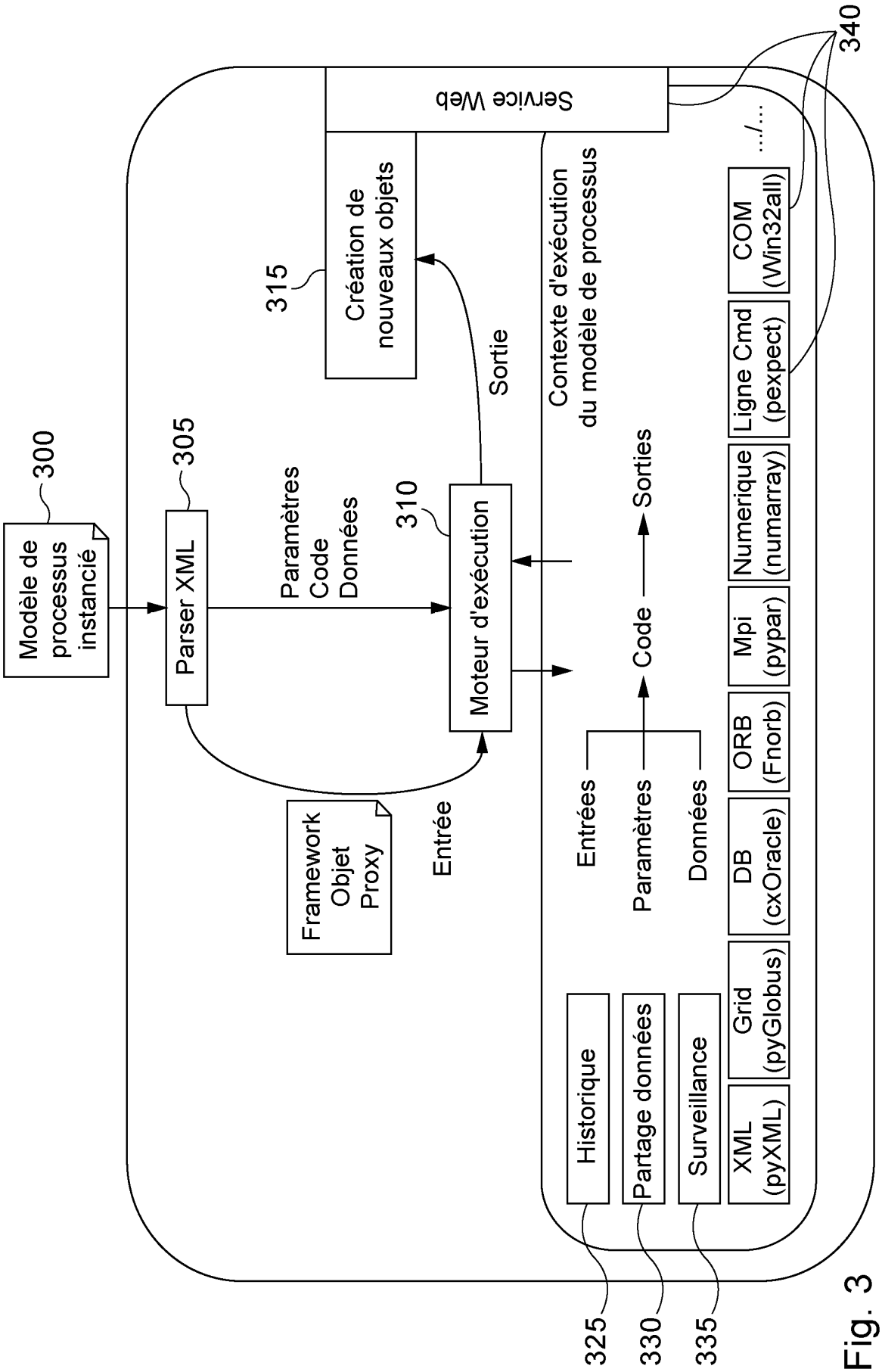

Modèle de processus instancié
300
Parser XML
305
Paramètres
Code
Données
Framework Objet Proxy
Entrée
Moteur d'exécution
310
Sortie
Création de nouveaux objets
315
Service Web
Contexte d'exécution du modèle de processus
Entrées
Paramètres
Données
Code
Sorties
Historique
Partage données
Surveillance
XML (pyXML)
Grid (pyGlobus)
DB (cxOracle)
ORB (Fnorb)
Mpi (pypar)
Numerique (numarray)
Ligne Cmd (pexpect)
COM (Win32all)
.../...
340
325
330
335

Fig. 3

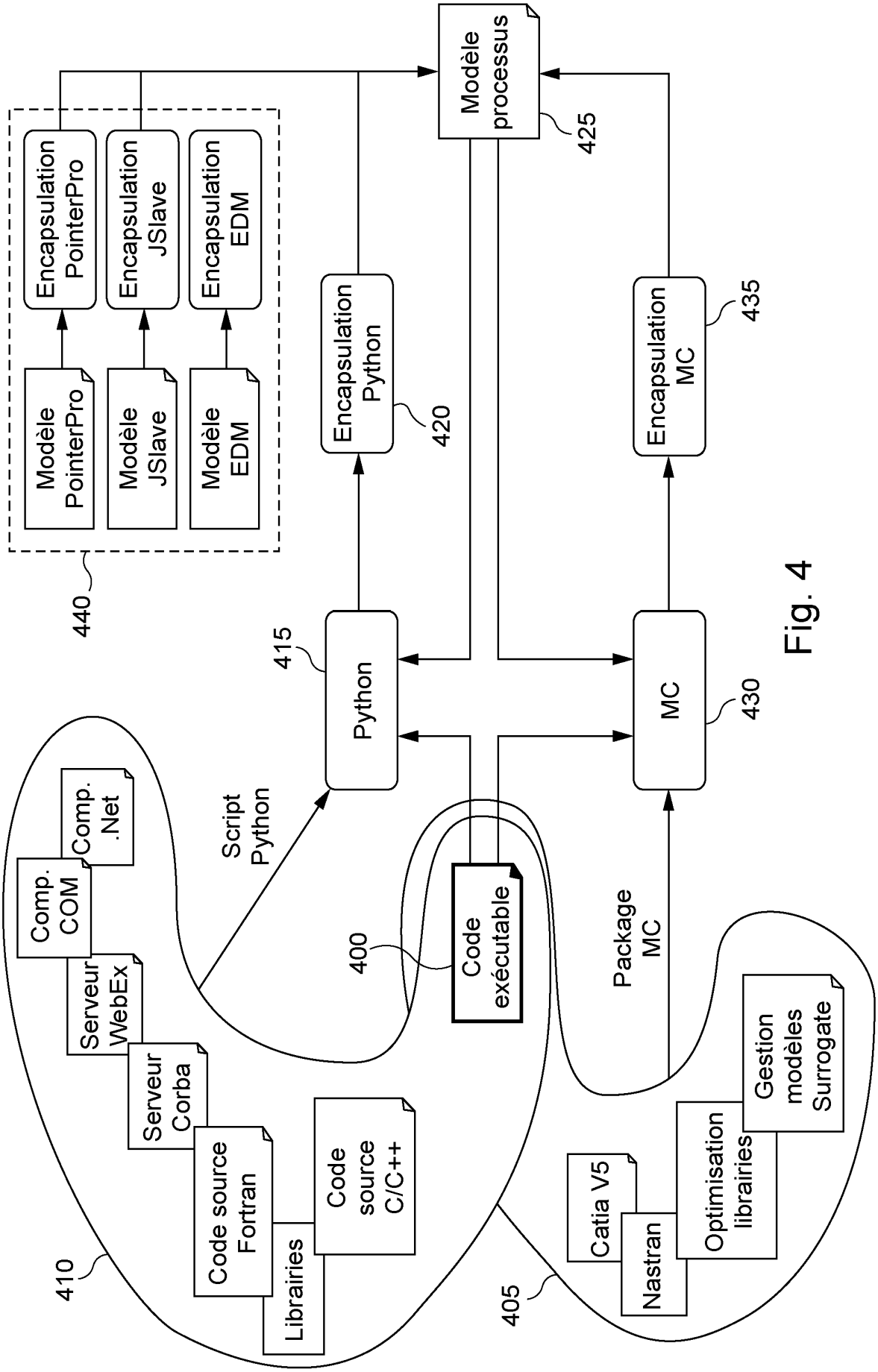

Modèle processus
425
Encapsulation PointerPro
Encapsulation JSlave
Encapsulation EDM
Modèle PointerPro
Modèle JSlave
Modèle EDM
440
Encapsulation Python
420
Encapsulation MC
435
415
Python
MC
430
Script Python
Package MC
400
Code exécutable
Comp. .Net
Comp. COM
Serveur WebEx
Serveur Corba
Code source Fortran
Code source C/C++
Librairies
410
Catia V5
Nastran
Optimisation librairies
Gestion modèles Surrogate
405

Fig. 4

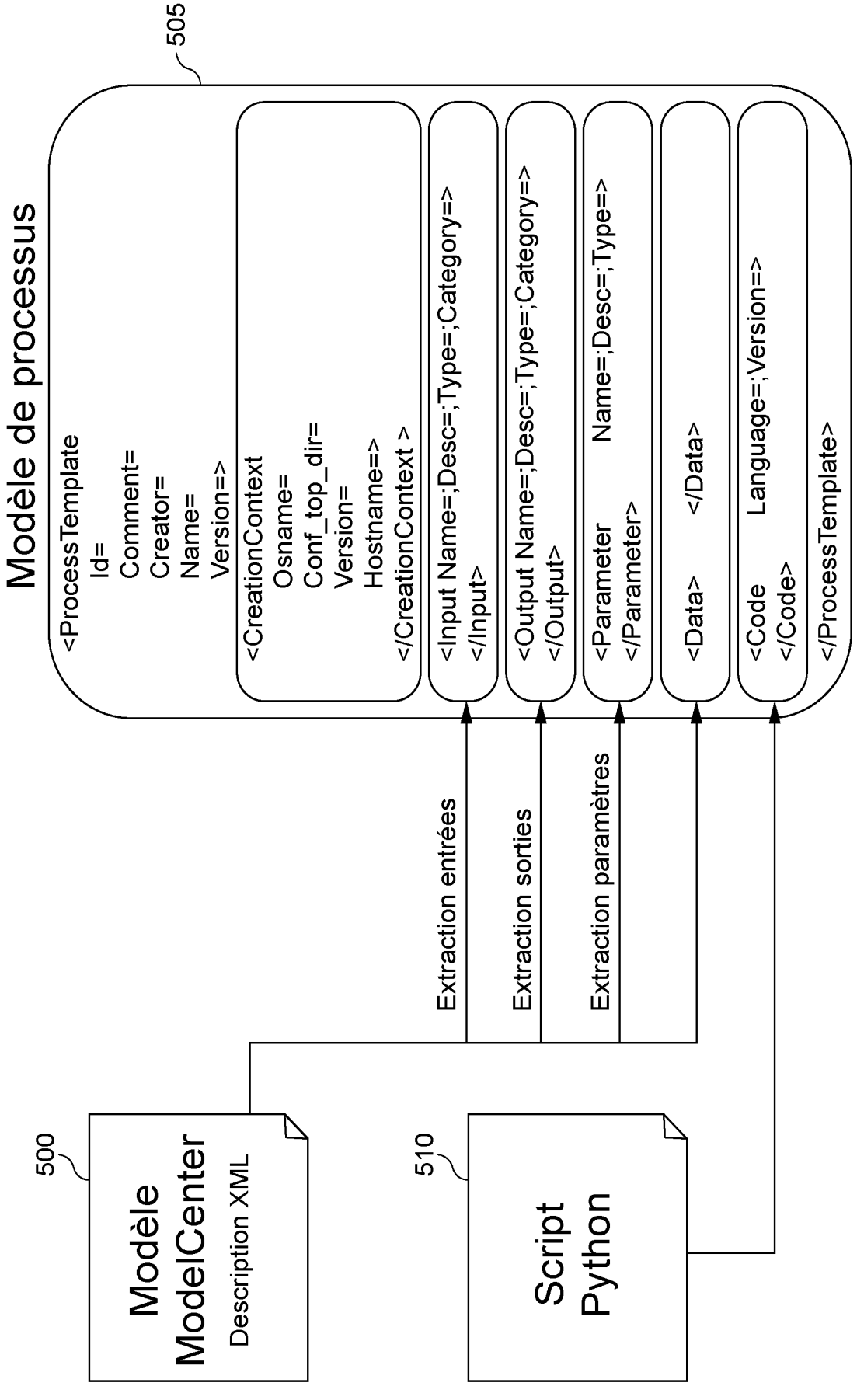

Modèle de processus
505
<ProcessTemplate
Id=
Comment=
Creator=
Name=
Version=>
<CreationContext
Osname=
Conf_top_dir=
Version=
Hostname=>
</CreationContext >
<Input Name=;Desc=;Type=;Category=>
</Input>
<Output Name=;Desc=;Type=;Category=>
</Output>
<Parameter Name=;Desc=;Type=>
</Parameter>
<Data> </Data>
<Code Language=;Version=>
</Code>
</ProcessTemplate>
Extraction entrées
Extraction sorties
Extraction paramètres
500
Modèle
ModelCenter
Description XML
510
Script
Python

Fig. 5

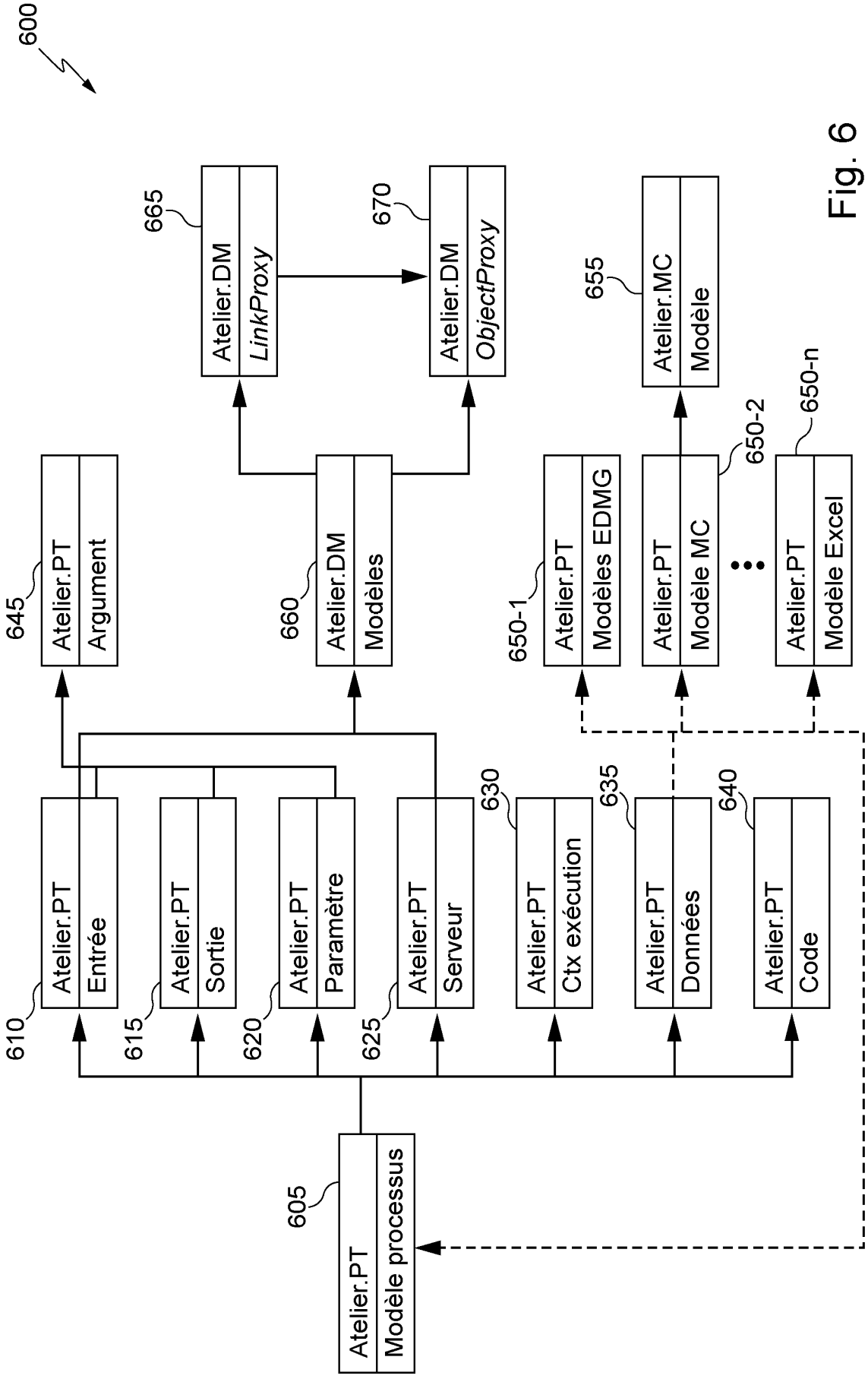

600
665
Atelier.DM
LinkProxy
670
Atelier.DM
ObjectProxy
655
Atelier.MC
Modèle
645
Atelier.PT
Argument
660
Atelier.DM
Modèles
650-1
Atelier.PT
Modèles EDMG
650-2
Atelier.PT
Modèle MC
650-n
Atelier.PT
Modèle Excel
610
Atelier.PT
Entrée
615
Atelier.PT
Sortie
620
Atelier.PT
Paramètre
625
Atelier.PT
Serveur
630
Atelier.PT
Ctx exécution
635
Atelier.PT
Données
640
Atelier.PT
Code
605
Atelier.PT
Modèle processus

Fig. 6

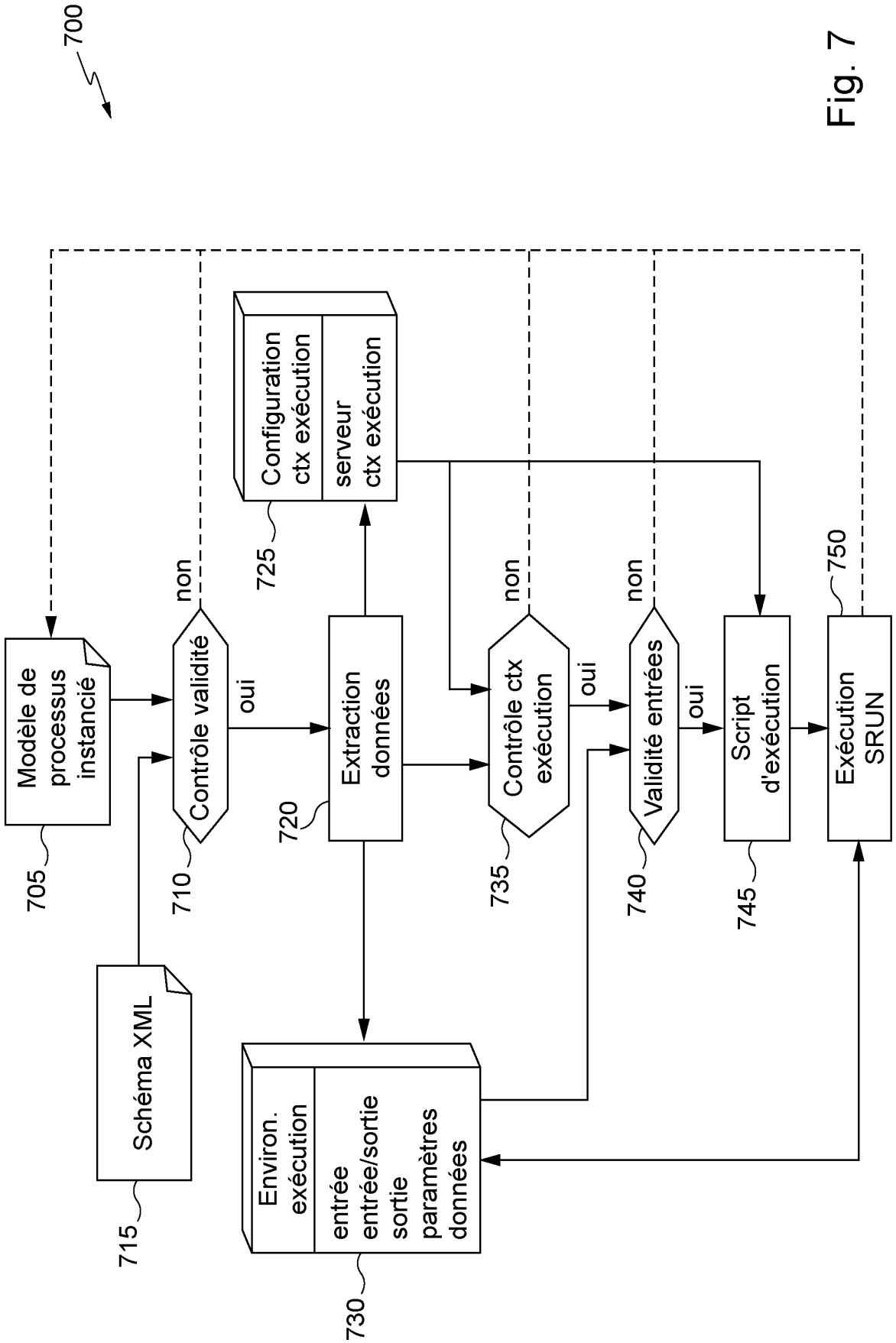

700
Modèle de processus instancié
705
Contrôle validité
non
oui
710
Schéma XML
715
Configuration ctx exécution
serveur ctx exécution
725
Extraction données
720
Environ. exécution
entrée
entrée/sortie
sortie
paramètres
données
730
Contrôle ctx exécution
non
oui
735
Validité entrées
non
oui
740
Script d'exécution
745
Exécution SRUN
750

Fig. 7

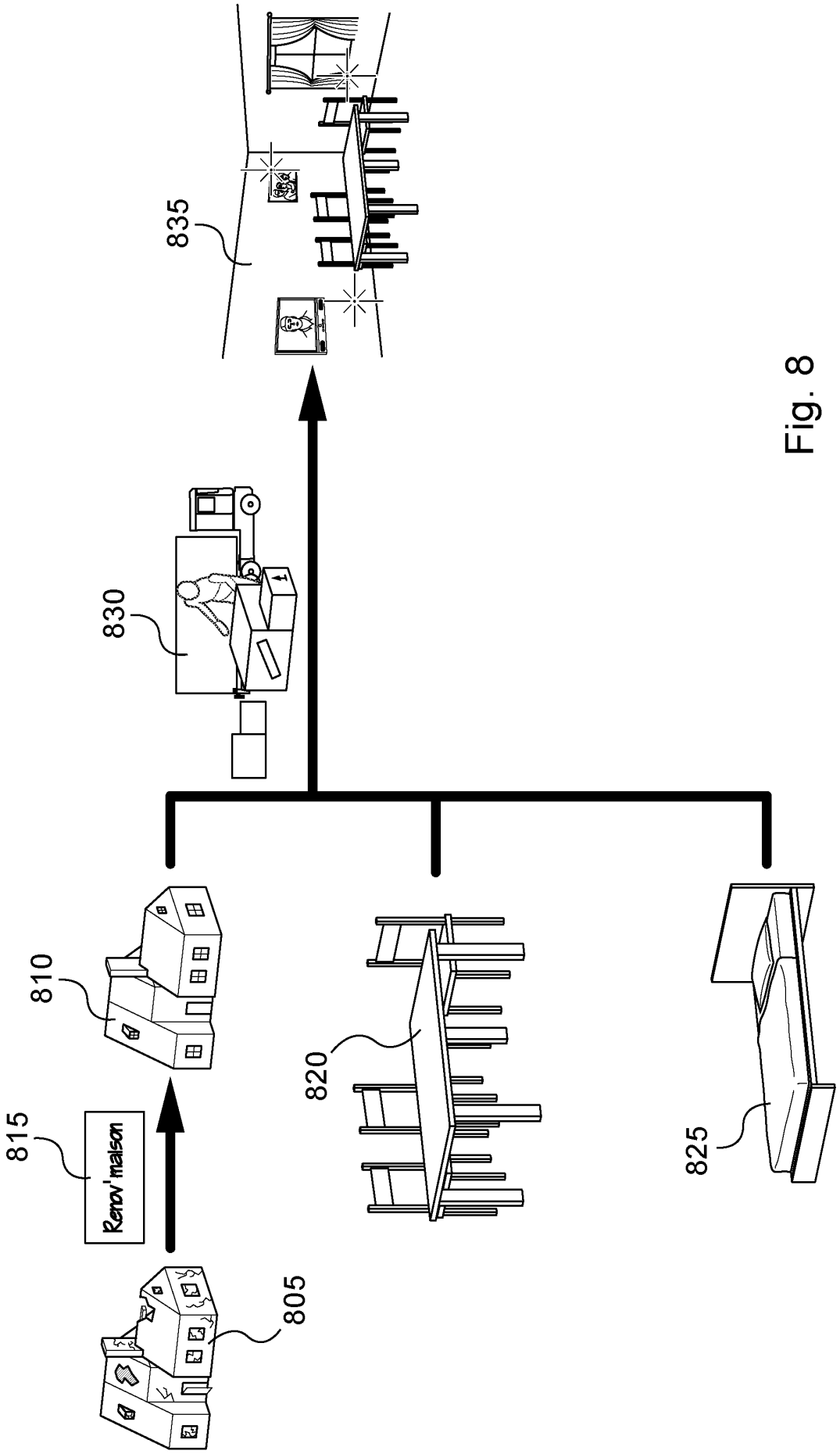


Fig. 8

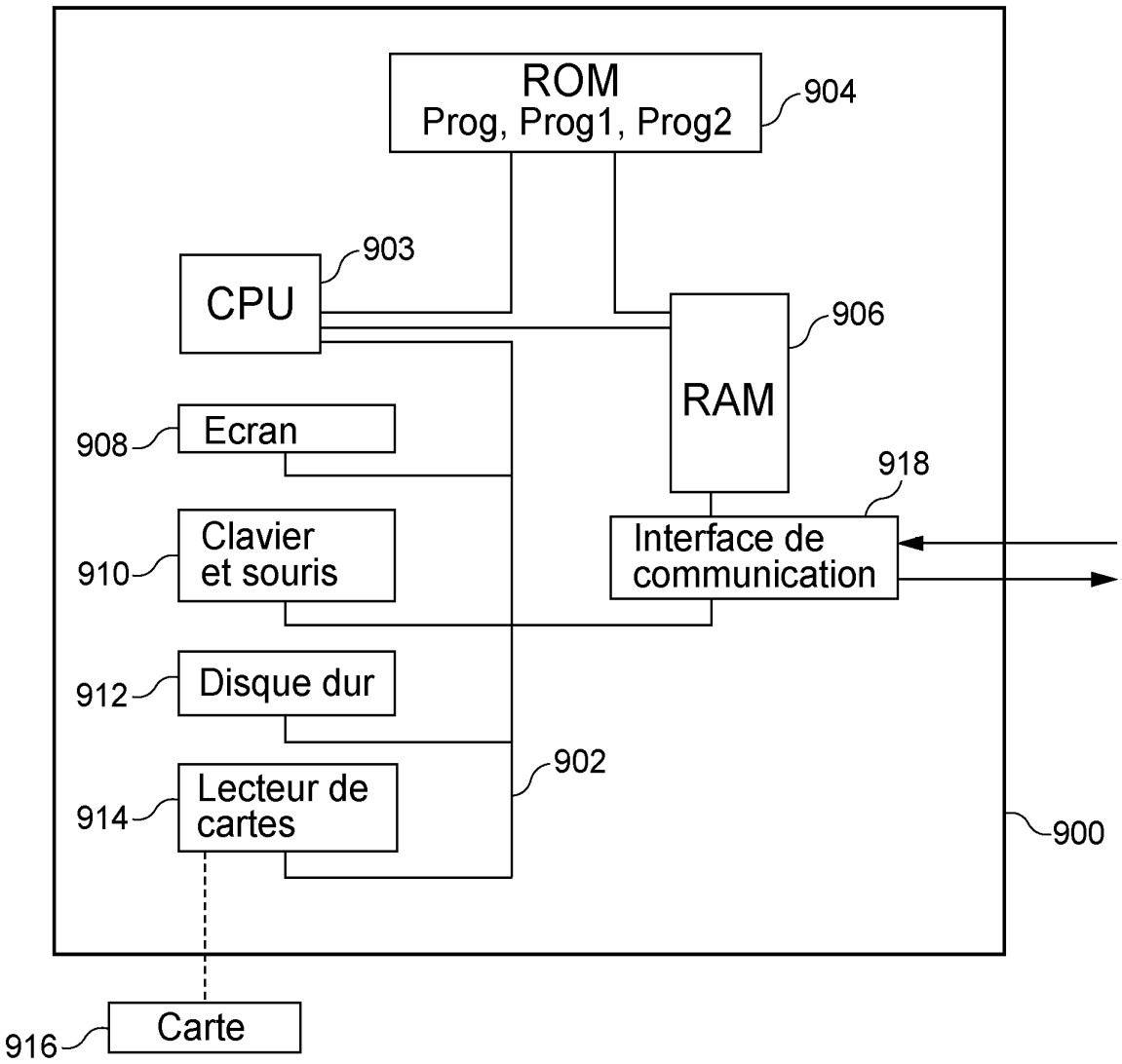

ROM
Prog, Prog1, Prog2
904
CPU
903
RAM
906
908
Ecran
910
Clavier
et souris
912
Disque dur
914
Lecteur de
cartes
902
918
Interface de
communication
900
916
Carte

Fig. 9

RÉPUBLIQUE FRANÇAISE

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 709757
FR 0853119

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		
X	NI LI ET AL: "Research and Realization of Collaborative M&S Services in Simulation Grid" CLUSTER COMPUTING AND THE GRID, 2006. CCGRID 06. SIXTH IEEE INTERNATIONAL SYMPOSIUM ON - SINGAPORE 16-19 MAY 2006, PISCATAWAY, NJ, USA,IEEE, vol. 2, 16 mai 2006 (2006-05-16), pages 69-69, XP010918279 ISBN: 978-0-7695-2585-3 * le document en entier * * figures 2,3,11,14 * -----	1-10	G06F17/50
X	ARDAIZ-VILLANUEVA O: "On-demand virtual studios and laboratories based on grid tecbnologies for project-based cooperative design and engineering" COMPUTER SUPPORTED COOPERATIVE WORK IN DESIGN, 2005. PROCEEDINGS OF TH E NINTH INTERNATIONAL CONFERENCE ON COVENTRY, UK MAY 24-26, 2005, PISCATAWAY, NJ, USA,IEEE, vol. 1, 24 mai 2005 (2005-05-24), pages 356-361, XP010832913 ISBN: 978-1-84600-002-7 * le document en entier * -----	1-10	DOMAINES TECHNIQUES RECHERCHÉS (IPC) G06Q G06F
X	LI QI ET AL: "Provisioning for Dynamic Instantiation of Community Services" IEEE INTERNET COMPUTING, IEEE SERVICE CENTER, NEW YORK, NY, US, vol. 11, no. 2, 1 mars 2008 (2008-03-01), pages 29-36, XP011205348 ISSN: 1089-7801 * page 33; figure 2 * * le document en entier * ----- -/--	1-10	

4

Date d'achèvement de la recherche	Examinateur
9 janvier 2009	Bauer, Rodolphe

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

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 709757
FR 0853119

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	US 2007/174697 A1 (SARIDAKIS TITOS [FI] ET AL) 26 juillet 2007 (2007-07-26) * le document en entier * -----		
A	ZHONGHUA YANG ET AL: "Automating integration of manufacturing systems and services: a semantic web services approach" INDUSTRIAL ELECTRONICS SOCIETY, 2005. IECON 2005. 31ST ANNUAL CONFERENCE OF IEEE, IEEE, PISCATAWAY, NJ, USA, 6 novembre 2005 (2005-11-06), pages 2249-2254, XP010876261 ISBN: 978-0-7803-9252-6 * le document en entier * * figure 5 * * page 2259 * -----	1-10	
A	FUJITSU: "Scaling processes and data flow across the global enterprise" INTERNET CITATION, [Online] 1 octobre 2007 (2007-10-01), pages 1-10, XP007906483 Extrait de l'Internet: URL:http://www.fujitsu.com/downloads/EU/uk/services/synfiniway/synfiniway whitepaper-scaling-processes-and-data-flow.pdf> [extrait le 2008-11-27] * le document en entier * -----	1-10	DOMAINES TECHNIQUES RECHERCHÉS (IPC)

Date d'achèvement de la recherche	Examinateur
9 janvier 2009	Bauer, Rodolphe

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

4

EPO FORM 1503 12.99 (P04C14)

ANNEXE AU RAPPORT DE RECHERCHE PRÉLIMINAIRE RELATIF A LA DEMANDE DE BREVET FRANÇAIS NO. FR 0853119 FA 709757

La présente annexe indique les membres de la famille de brevets relatifs aux documents brevets cités dans le rapport de recherche préliminaire visé ci-dessus.
Les dits membres sont contenus au fichier informatique de l'Office européen des brevets à la date du 09-01-2009
Les renseignements fournis sont donnés à titre indicatif et n'engagent pas la responsabilité de l'Office européen des brevets, ni de l'Administration française

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 2007174697 A1	26-07-2007	AUCUN	

EPO FORM P0465