(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
29 décembre 2016 (29.12.2016)

WIPO | PCT

(10) Numéro de publication internationale
WO 2016/207520 A1

(51) Classification internationale des brevets :
*G06F 9/445* (2006.01) *G06F 9/50* (2006.01)
*G06F 9/455* (2006.01) *H04L 29/08* (2006.01)

(21) Numéro de la demande internationale :
PCT/FR2016/051466

(22) Date de dépôt international :
16 juin 2016 (16.06.2016)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
1555936 26 juin 2015 (26.06.2015) FR

(71) Déposant : **ORANGE** [FR/FR]; 78 rue Olivier de Serres, 75015 Paris (FR).

(72) Inventeurs : **ETCHEVERS, Xavier**; 20 avenue Marius Cottier, 38700 Corenc (FR). **COUPAYE, Thierry**; 368 rue Saint-Sulpice, 38920 Crolles (FR).

(74) Mandataire : **ORANGE/IPL**; LECOMTE Isabelle, 38-40 rue du Général Leclerc, 92794 Issy Moulineaux Cedex 9 (FR).

(81) États désignés *(sauf indication contraire, pour tout titre de protection nationale disponible)* : AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) États désignés *(sauf indication contraire, pour tout titre de protection régionale disponible)* : ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), eurasien (AM, AZ, BY, KG, KZ, RU, TJ, TM), européen (AL, AT, BE, BG, CH, CY, CZ, DE,

*[Suite sur la page suivante]*

(54) Title : TECHNIQUE FOR RECONFIGURING AN APPLICATION MODEL FOR A DEPLOYMENT OF A SOFTWARE APPLICATION

(54) Titre : TECHNIQUE DE RECONFIGURATION D'UN MODELE D'APPLICATION POUR UN DEPLOIEMENT D'UNE APPLICATION LOGICIELLE

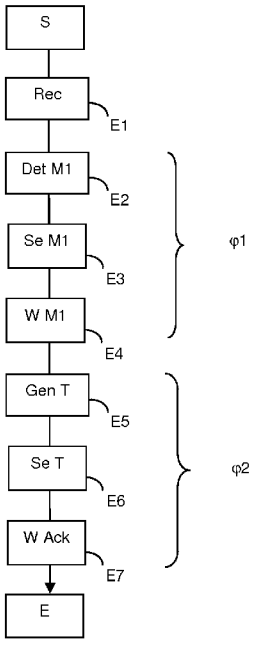



Fig. 3

(57) Abstract : The invention relates to a technique for reconfiguring an application model for a deployment of a software application. An application model comprises at least one component and deployment constraints. A current application model, called the executing model, is deployed in an infrastructure of virtual machines. The manager sends to a configurator responsible for a component a command to deploy an intermediate model comprising said at least one component of the executing model and in which one component present in the executing model and absent from a target application model is deleted, and one component present in the executing model and stopped in the target application model is indicated as stopped. Once the intermediate model has been deployed to the infrastructure, the manager then sends to a configurator responsible for a component a command to deploy the target application model. The target application model becomes the executing model once deployed to the infrastructure.

(57) Abrégé : L'invention concerne une technique de reconfiguration d'un modèle d'application pour un déploiement d'une application logicielle. Un modèle d'application comprend au moins un composant applicatif et des contraintes de déploiement. Un modèle d'application courant, dit modèle à l'exécution, est déployé sur une infrastructure

*[Suite sur la page suivante]*

WO 2016/207520 A1

DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Publiée** :

— *avec rapport de recherche internationale (Art. 21(3))*

de machines virtuelles. Le gestionnaire envoie à un configurateur responsable d'un composant applicatif une commande de déploiement d'un modèle intermédiaire, comprenant ledit au moins un composant du modèle à l'exécution et dans lequel, un composant présent dans le modèle à l'exécution et absent d'un modèle d'application cible est supprimé, et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter. Lorsque le modèle intermédiaire est déployé sur l'infrastructure, le gestionnaire envoie alors à un configurateur responsable d'un composant applicatif une commande de déploiement du modèle d'application cible. Le modèle d'application cible devient le modèle à l'exécution une fois déployé sur l'infrastructure.

# Technique de reconfiguration d'un modèle d'application pour un déploiement d'une application logicielle

L'invention se rapporte au domaine général des applications logicielles.

Elle concerne plus particulièrement une technique de reconfiguration d'une application logicielle déployée dans un environnement d'exécution.

L'invention s'applique à des applications logicielles déployées dans un système informatique dans le nuage aussi plus communément appelé « cloud » en anglais.

Les environnements de type informatique dans le nuage se déclinent en trois grands niveaux d'offre selon le type de ressource mis à disposition. Le niveau « Infrastructure as a Service » (IaaS) a pour but de permettre d'accéder à des ressources matérielles (calcul, stockage, réseau) virtualisées s'appuyant sur un ensemble de ressources matérielles physiques. La couche « Software as a Service » (SaaS) vise à exposer des applications logicielles à destination des utilisateurs finaux. La couche intermédiaire « Platform as a Service » (PaaS) offre un ensemble d'outils et d'environnements d'exécution qui permettent de gérer le cycle de vie des applications. Ce cycle de vie comprend notamment les phases de conception, développement, déploiement et plus généralement d'administration des applications (gestion de la charge, de la tolérance aux pannes, de la sécurité, etc.).

Pour déployer une application logicielle dans un environnement virtualisé de type informatique dans le nuage, il est nécessaire de générer des images virtuelles qui vont être instanciées sous forme de machines virtuelles pour supporter l'exécution de l'application sur une plate-forme de type IaaS. Chaque image logicielle se compose de briques techniques (système d'exploitation et intergiciels) et de briques fonctionnelles (données et logiciels applicatifs). Une fois instanciée, chaque machine virtuelle fait généralement l'objet d'une phase de configuration dynamique finalisant le paramétrage de l'application répartie.

Une ressource virtualisée désigne un artefact logiciel dont la finalité est la dématérialisation d'une ressource physique (ou réelle). Pour cela, elle en reproduit les comportements et les caractéristiques à l'identique. Ainsi, de manière comparable à une machine physique, une machine virtuelle est caractérisée par le nombre de processeurs, la quantité de mémoire, les unités de stockage et les interfaces réseaux dont elle dispose. En outre, elle fonctionne de façon identique à n'importe quelle machine réelle. La virtualisation des ressources physiques ne se limite pas aux serveurs et concerne également les entités de stockage et les équipements réseaux.

L'un des principaux intérêts de la virtualisation est de permettre la consolidation des ressources matérielles par mutualisation. Cela consiste à mettre en œuvre simultanément un ensemble de ressources matérielles virtualisées au niveau d'une infrastructure matérielle physique commune (i.e. plusieurs machines virtuelles s'exécutant sur une même machine physique).

L'architecture d'une application correspond à la modélisation, d'une part, de l'ensemble des éléments logiciels (ex. exécutables et données) et matériels (ex. machines) nécessaires au fonctionnement de l'application et, d'autre part, à l'ensemble des contraintes (ou dépendances) qui lient ces éléments entre eux. La nature de ces dépendances est variable. Elles peuvent notamment concerner des dépendances :

- de placement d'un élément logiciel par rapport à un autre (i.e. co-localisation, non-co-localisation au sein d'une machine virtuelle) ;
- de configuration entre deux éléments logiciels;
- d'activation / désactivation d'un élément logiciel par rapport à un autre.

On se place par la suite dans le cadre d'un modèle d'application reposant sur des composants permettant de décrire les éléments logiciels qui constituent une application et leur projection sur une infrastructure d'exécution ainsi que les dépendances qui les lient au sein de l'architecture applicative. Une application logicielle est ainsi modélisée par un modèle d'application comprenant au moins un composant administrant un élément logiciel et des contraintes de déploiement.

De façon générale, reconfigurer une application consiste à faire évoluer son architecture. La reconfiguration comprend des opérations élémentaires d'ajout, de suppression et/ou de modification des éléments logiciels et des dépendances entre éléments qui composent l'architecture applicative.

Dans une approche décentralisée, où les configurations des machines virtuelles s'exécutent indépendamment les unes des autres, il existe un risque d'apparition d'incohérences dans les états des composants administrant les éléments logiciels lors de la reconfiguration de l'application. A titre d'exemple illustratif, dans une nouvelle configuration, un composant A doit se connecter à un composant à créer B sur une autre machine virtuelle. Lorsque le composant A essaie de se connecter, le composant B n'est pas encore créé. La reconfiguration de l'application échoue.

Un des buts de l'invention est de remédier à des insuffisances/inconvénients de l'état de la technique et/ou d'y apporter des améliorations.

Selon un premier aspect, l'invention a pour objet un procédé de reconfiguration d'un modèle d'application pour un déploiement d'une application logicielle. Un modèle d'application comprend au moins un composant applicatif et des contraintes de déploiement et un modèle d'application courant, dit modèle à l'exécution, est déployé sur une infrastructure de machines virtuelles reposant sur au moins une machine physique. Le procédé comprend :

- obtention par un gestionnaire de déploiement d'un nouveau modèle d'application, dit modèle d'application cible ;
- envoi, par le gestionnaire de déploiement à un configurateur responsable d'un composant applicatif, d'une commande de déploiement d'un modèle intermédiaire, comprenant

ledit au moins un composant du modèle à l'exécution et dans lequel, un composant présent dans le modèle à l'exécution et absent du modèle d'application cible est supprimé, et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter ;

- lorsque le modèle intermédiaire est déployé sur l'infrastructure, envoi par le gestionnaire de déploiement à un configurateur responsable d'un composant applicatif, d'une commande de déploiement du modèle d'application cible,

le modèle d'application cible devenant le modèle à l'exécution une fois déployé sur l'infrastructure.

Plus précisément, le modèle intermédiaire envoyé est déterminé par le gestionnaire de déploiement.

Ainsi, par la décomposition de la reconfiguration en deux phases, il est possible de gérer correctement les états des composants, sans risque d'apparition d'incohérence. Une synchronisation est introduite à la fin de la première phase, une fois le modèle intermédiaire déployé.

A partir de la description exhaustive d'une architecture applicative cible fournie par l'administrateur de l'application, il est possible de mettre en œuvre, en deux phases, les actions de reconfiguration permettant de faire évoluer le modèle courant de l'application, dit modèle à l'exécution, vers le nouveau modèle. Pour cela, la solution de reconfiguration s'appuie sur le modèle de l'application courant, le modèle cible de l'application à mettre en place et un protocole en deux phases asynchrones, décentralisé et fiable, capable de faire évoluer l'instance courante de l'application vers l'architecture cible tout en respectant les dépendances associées (i.e. de configuration, de placement et d'activation). La première phase de ce protocole se concentre sur les opérations de suppression et d'arrêt de composants applicatifs. La seconde phase vise à procéder aux opérations d'ajout, de configuration et d'activation de composants applicatifs. Le déclenchement de la seconde phase est conditionné par la finalisation complète de la première grâce à un point de synchronisation prévu en fin de première phase.

On bénéficie ainsi des avantages associés à la mise en œuvre d'une méthode décentralisée et asynchrone. Il est ainsi possible de reconfigurer tout type d'application, indépendamment de son architecture, des technologies sur lesquelles elle s'appuie ou du domaine métier qu'elle prend en charge. Il est également possible de reconfigurer tout type d'application, indépendamment de l'infrastructure d'exécution (e.g. couche IaaS) sur laquelle est instanciée l'application. Les reconfigurations ne se restreignent pas à une reconfiguration d'une machine virtuelle mais permettent également de modifier n'importe quel sous-ensemble applicatif, sans restriction de nature ou de taille. La technique de reconfiguration permet également de mener à son terme, dans un temps fini et en présence d'un nombre fini de pannes franches de l'infrastructure d'exécution de l'application (par exemple les machines virtuelles) une opération de reconfiguration. La cohérence architecturale est par ailleurs maintenue du fait du respect des dépendances notamment d'activation entre les différents éléments.

La technique de reconfiguration permet d'adapter aux exigences de l'utilisateur la quantité et la configuration des ressources matérielles et logicielles nécessaires à la mise en œuvre de l'application.

Dans un mode de réalisation particulier, le procédé de reconfiguration comprend préalablement à l'envoi du modèle d'application cible, lorsqu'un composant nécessite la mise en œuvre d'un élément logiciel non publié dans une image logicielle, une génération d'au moins une nouvelle image logicielle nécessaire à la mise en œuvre du nouveau composant, publication et instanciation sous forme d'une machine virtuelle de l'infrastructure.

Ainsi, un nouveau composant peut être créé.

Dans un mode de réalisation particulier, une estampille ordonnée est associée à un modèle d'application envoyé à un configurateur.

L'estampillage ordonné du modèle d'application envoyé permet de procéder à des synchronisations locales entre deux configurateurs. Ceci est particulièrement avantageux en termes de performance et de vitesse d'exécution par rapport à des modes de fonctionnement où le gestionnaire de déploiement centralise et synchronise l'exécution des opérations de reconfiguration. Ce mécanisme d'estampillage permet de maintenir l'exécution en parallèle, de façon asynchrone, des configurateurs.

Selon un deuxième aspect, au niveau d'un configurateur responsable d'un composant, le procédé de reconfiguration comprend :

- réception, par un configurateur responsable d'un composant applicatif en provenance d'un gestionnaire de déploiement, d'une commande de déploiement d'un modèle intermédiaire, comprenant ledit au moins un composant du modèle à l'exécution et dans lequel un composant présent dans le modèle à l'exécution et absent du modèle d'application cible est supprimé et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter ;
- exécution d'une action, ladite action appartenant à un groupe comprenant une suppression d'un composant lorsque ledit composant est absent du modèle intermédiaire, un arrêt d'un composant lorsque ledit composant est indiqué à arrêter, un arrêt du composant lorsque ledit composant comprend une interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant absent du modèle intermédiaire ;
- envoi au gestionnaire de déploiement d'une notification de changement d'état du composant ;
- réception, par un configurateur responsable d'un composant applicatif en provenance d'un gestionnaire de déploiement, d'une commande de déploiement du modèle d'application cible ;
- exécution d'une action, ladite action appartenant à un groupe comprenant une création d'un nouveau composant, une modification d'un composant, une activation d'un composant arrêté

lorsque ledit composant comprend une interface serveur reliée par une liaison obligatoire à une interface client d'un nouveau composant ;

- envoi au gestionnaire de déploiement d'une notification de changement d'état du composant.

Les avantages énoncés pour le procédé de reconfiguration selon le premier aspect sont également applicables lorsque ce procédé est mis en œuvre par un configurateur.

Dans un mode de réalisation particulier, un message envoyé à un configurateur responsable d'un composant lié comprend une estampille reçue avec un modèle d'application.

Les configurateurs peuvent ainsi être synchronisés deux à deux sans nécessiter de synchronisation centralisée.

Dans un mode de réalisation particulier, une action de suppression ou d'arrêt d'un composant applicatif est effectuée lorsque ledit composant n'a pas d'interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant applicatif.

Un composant applicatif ne peut pas s'arrêter ou être supprimé tant qu'il a une interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant applicatif. Ceci permet de garantir que cet autre composant applicatif ne va pas continuer à s'exécuter sans recevoir les données nécessaires à son exécution. La cohérence architecturale de l'application est ainsi maintenue grâce au respect des dépendances.

Dans un mode de réalisation particulier, lors de l'exécution de l'action de suppression ou d'arrêt d'un composant applicatif, le configurateur responsable dudit composant envoie un message d'arrêt de notification pour une interface client reliée par une liaison à une interface serveur d'un autre composant.

Un composant applicatif lorsqu'il s'arrête ou lorsqu'il va être supprimé signale cet arrêt pour une interface client reliée à une interface serveur d'un autre composant. Ceci permet à cet autre composant de s'arrêter sans créer de disfonctionnement dans l'exécution de l'application. La cohérence architecturale de l'application est ainsi maintenue grâce au respect des dépendances.

Selon un troisième aspect, l'invention concerne en outre un dispositif mettant en œuvre un gestionnaire de déploiement s'exécutant sur une machine virtuelle hébergée par le dispositif, ledit gestionnaire de déploiement comprenant pour reconfigurer un modèle d'application pour un déploiement d'une application logicielle, un modèle d'application comprenant au moins un composant applicatif et des contraintes de déploiement, un modèle d'application courant, dit modèle à l'exécution, étant déployé sur une infrastructure de machines virtuelles reposant sur au moins une machine physique :

- un module d'obtention d'un nouveau modèle d'application, dit modèle d'application cible ;

- un module de détermination d'un modèle intermédiaire, comprenant ledit au moins un composant du modèle à l'exécution et dans lequel un composant présent dans le modèle à l'exécution et absent du modèle d'application cible est supprimé et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter ;

- un module d'envoi à un configurateur responsable d'un composant applicatif, d'une commande de déploiement d'un modèle à appliquer ;

- un module de génération d'au moins une nouvelle image logicielle nécessaire à la mise en œuvre d'un nouveau composant, et d'instanciation sous forme d'une machine virtuelle de l'infrastructure ;

- un module de commande, agencé pour commander le module d'envoi pour envoyer le modèle intermédiaire déterminé et lorsque le modèle intermédiaire est déployé sur l'infrastructure, pour activer le module de génération/instanciation et pour commander le module d'envoi pour envoyer le modèle d'application cible.

Les avantages énoncés pour le procédé de reconfiguration selon le premier aspect sont transposables directement au dispositif mettant en œuvre un gestionnaire de déploiement.

Dans un mode de réalisation particulier du dispositif mettant en œuvre un gestionnaire de déploiement, le module de génération est activé pour générer une nouvelle image logicielle lorsqu'un composant nécessite la mise en œuvre d'un élément logiciel non publié dans une image logicielle.

Selon un quatrième aspect, l'invention concerne également un dispositif mettant en œuvre un configurateur s'exécutant sur une machine virtuelle hébergée par le dispositif, ledit configurateur comprenant pour reconfigurer un modèle d'application pour un déploiement d'une application logicielle, un modèle d'application comprenant au moins un composant applicatif et des contraintes de déploiement, un modèle d'application courant, dit modèle à l'exécution, étant déployé sur une infrastructure de machines virtuelles reposant sur au moins une machine physique :

- un module de réception en provenance d'un gestionnaire de déploiement, d'une commande de déploiement d'un modèle à appliquer, ce modèle correspondant au modèle cible ou à un modèle intermédiaire comprenant ledit au moins un composant du modèle à l'exécution et dans lequel, un composant présent dans le modèle à l'exécution et absent du modèle d'application cible est supprimé, et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter ;

- un module d'exécution d'une action, ladite action appartenant à un groupe comprenant une suppression d'un composant lorsque ledit composant est absent du modèle à appliquer, un arrêt d'un composant lorsque ledit composant est indiqué à arrêter, un arrêt d'un composant lorsque ledit composant comprend une interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant absent du modèle à appliquer, une création d'un nouveau composant, une

modification d'un composant, une activation d'un composant arrêté lorsque ledit composant comprend une interface serveur reliée par une liaison obligatoire à une interface client d'un nouveau composant ;

- un module d'envoi au gestionnaire de déploiement d'une notification de changement d'état du composant.

Les avantages énoncés pour le procédé de reconfiguration selon le deuxième aspect sont transposables directement au dispositif mettant en œuvre un configurateur.

Selon un cinquième aspect, l'invention concerne en outre un programme pour un dispositif, comprenant des instructions de code de programme destinées à commander l'exécution des étapes du procédé de reconfiguration selon le premier aspect mises en œuvre par le dispositif, lorsque ce programme est exécuté par ce dispositif et un support d'enregistrement lisible par un dispositif sur lequel est enregistré un programme pour un dispositif.

Les avantages énoncés pour le procédé de reconfiguration selon le premier aspect sont transposables directement au programme pour un dispositif et au support d'enregistrement.

Selon un sixième aspect, l'invention concerne en outre un programme pour un dispositif, comprenant des instructions de code de programme destinées à commander l'exécution des étapes du procédé de reconfiguration selon le deuxième aspect mises en œuvre par le dispositif, lorsque ce programme est exécuté par ce dispositif et un support d'enregistrement lisible par un dispositif sur lequel est enregistré un programme pour un dispositif.

Les avantages énoncés pour le procédé de reconfiguration selon le deuxième aspect sont transposables directement au programme pour un dispositif et au support d'enregistrement.

La technique de reconfiguration sera mieux comprise à l'aide de la description suivante de modes de réalisation particuliers, en référence aux dessins annexés sur lesquels :

- la figure 1 représente un environnement dans lequel est mis en œuvre le procédé de de reconfiguration dans un mode de réalisation particulier ;
- la figure 2a représente schématiquement un premier exemple d'une application logicielle déployée selon un mode particulier de réalisation ;
- la figure 2b représente schématiquement un deuxième exemple d'une application logicielle déployée selon un mode particulier de réalisation ;
- la figure 3 illustre des étapes d'un procédé de reconfiguration mises en œuvre par un gestionnaire de déploiement selon un mode particulier de réalisation ;
- les figures 4a et 4b illustrent des étapes d'un procédé de reconfiguration mises en œuvre par un configurateur selon un mode particulier de réalisation ;
- la figure 5 représente schématiquement un dispositif mettant en œuvre un gestionnaire de déploiement dans un mode particulier de réalisation ;

- la figure 6 représente schématiquement un dispositif mettant en œuvre un configurateur dans un mode particulier de réalisation.

La **figure 1** représente, dans son environnement, un système 200 de gestion d'une configuration de machines pour le déploiement d'une application logicielle APP dans un mode particulier de réalisation. Le système 200 permet de gérer des ressources virtuelles nécessaires à l'exécution d'une application logicielle APP. La configuration et le dimensionnement de ces ressources virtuelles ont été déterminées préalablement. Une fois cette configuration déployée, l'application logicielle peut être exécutée dans un environnement réel d'exécution.

Aucune limitation n'est attachée à la nature de l'application logicielle APP. Il peut s'agir d'une application permettant l'accès à ou la génération de contenus multimédia, le référencement de produits, l'accès à des équipements distants, etc.

Dans l'exemple envisagé ici, l'application logicielle APP est destinée à être déployée (i.e. mise en production) dans un système informatique en nuage ou cloud (non représenté), sur une configuration de machines (ou serveurs) virtuelles hébergées par le cloud. Les ressources partagées peuvent être de différentes natures : il peut s'agir notamment de ressources de type processeur ou CPU de machines (ou serveurs) virtuelles et/ou physiques, de ressources mémoires (ex. RAM, disque, etc.), ou encore de ressources réseaux (ex. connecteurs réseaux, etc.).

Une application logicielle APP comprend un ensemble d'éléments logiciels 240 qui fournissent des services et qui interagissent entre eux pour produire les fonctionnalités de l'application dans un plan applicatif. Ces éléments logiciels sont répartis au sein d'une infrastructure d'exécution composée d'un ensemble de machines virtuelles 22 (notées VM pour « Virtual Machine »). Au-delà des éléments logiciels qui la composent, une application logicielle est également définie à l'aide des contraintes de déploiement (ou dépendances) qui lient ces éléments logiciels entre eux. Il existe trois types de dépendances :

- les dépendances de configuration : un élément A a une dépendance de configuration vis-à-vis d'un élément B, si l'élément A doit disposer d'informations relatives à l'élément B pour pouvoir être configuré et interagir avec lui (e. g. l'adresse IP de l'élément B, des identifiants de connexion à une base de données, etc.)
- les dépendances d'activation : un élément A a une dépendance d'activation vis-à-vis d'un élément B, si le démarrage de l'élément A est conditionné par celui de l'élément B. Dans un tel cas, dans la suite du document, une telle dépendance induit également la réciproque, à savoir qu'une telle dépendance implique que l'arrêt de l'élément B est conditionné par celui de l'élément A.
- les dépendances de placement : les éléments A et B ont une dépendance de placement, si les éléments A et B doivent être co-localisés (respectivement non co-localisés) au sein de l'infrastructure d'exécution (i. e. sur une même machine virtuelle).

On se place par la suite dans le cas particulier d'une application logicielle APP qui est mise en œuvre dans une plateforme VAMP (pour « Virtual Application Management Platform »). La plateforme VAMP est une structure logicielle (ou « framework » logiciel) qui permet un déploiement d'applications arbitraires dans un système informatique en nuage ou cloud sur une configuration de machines (ou serveurs) virtuelles hébergées par le cloud. VAMP permet d'assurer le déploiement autonome, générique et fiable de toute application répartie dans un environnement de type informatique dans le nuage. Une telle plateforme VAMP est décrite dans l'article « Configuration of Distributed Applications in the Cloud » de X. Etchevers et publié dans les actes de la conférence IEEE Cloud 2011, 668-675. Une application est dite arbitraire si aucune hypothèse n'est faite quant à son architecture (i. e. l'organisation des éléments logiciels qui la composent), des technologies sur lesquelles elle repose ou des domaines métiers qu'elle prend en charge. Par la suite, le terme application logicielle désigne une application arbitraire. Un déploiement initial de cette application logicielle APP a été effectué conformément à l'article « Automated Configuration of Legacy Applications in the Cloud » de X. Etchevers et al, publié dans les actes de la conférence UCC 2011, 170-177.

Une plateforme VAMP comprend :

- un modèle à base de composants permettant de décrire les éléments logiciels qui constituent une application et leur projection sur une infrastructure d'exécution ainsi que les dépendances qui les lient au sein de l'architecture applicative ;
- un protocole asynchrone, réparti et fiable d'auto-configuration et d'auto-activation de l'application ;
- des mécanismes visant à fiabiliser le système VAMP lui-même.

Une plateforme VAMP s'appuie sur un modèle d'application composé :

- d'un modèle d'architecture basée sur le modèle à composant Fractal (défini par le consortium OW2, Fractal ADL, disponible sur le site Web http://fractal.ow2.org/fractaladl/) et permettant de définir l'organisation d'une application logicielle répartie selon un ensemble d'unités d'exécution, et
- d'un modèle de machine virtuelle, qui modélise l'infrastructure matérielle et logicielle sur laquelle l'application va être projetée et qui introduit deux extensions au formalisme défini par la spécification OVF (pour « Open Virtualization Format »).

Une machine virtuelle est vue comme l'agrégation d'un ensemble de processeurs, d'une quantité de disques, d'interfaces réseaux et de disques. Chaque disque se caractérise par une taille et une image peut lui être associée. Le contenu d'une image est représenté au moyen d'un système d'exploitation, d'un ensemble d'intergiciels ( ou « middleware ») et de données utilisateur qui peuvent aussi bien être des binaires que des données applicatives. Un module de virtualisation ou hyperviseur désigne un environnement d'exécution capable de créer, activer, migrer, stopper et

détruire des machines virtuelles. L'hyperviseur gère l'allocation des ressources matérielles entre les différentes instances de machines virtuelles et met à disposition des machines virtuelles ces ressources virtualisées.

Un gestionnaire VAMP 100 constitue le point d'accès unique au service de déploiement. Tel que représenté sur la figure 1, il est installé sur une machine virtuelle 10. Aucune limitation n'est attachée à cette représentation, le gestionnaire VAMP pouvant également être installé sur une machine physique. Le gestionnaire VAMP 100 a pour principale mission de traiter les requêtes de déploiement d'application logicielle provenant des utilisateurs. Pour chaque nouvelle requête, il instancie dynamiquement un nouveau gestionnaire de déploiement 210 dans une nouvelle machine virtuelle 21 et il lui transmet la requête de déploiement associée. Outre ce rôle, le gestionnaire VAMP 100 permet également à un utilisateur d'obtenir un état d'avancement du déploiement de son application.

VAMP définit une architecture en couche regroupant un ensemble d'entités de contrôle situées dans un plan d'administration :

- les gestionnaires de déploiement qui procèdent à la mise en œuvre des phases du processus de déploiement comprises entre la génération des images et leur instanciation au sein de machines virtuelles. Dans un souci de fiabilisation et d'isolation des problématiques de déploiement, il existe un gestionnaire dédié au déploiement de chaque application.

- des agents de configuration ou configurateurs qui coopèrent entre eux pour parvenir à la finalisation du processus de déploiement (i.e. post-configuration et démarrage).

- des composants qui encapsulent les éléments logiciels qui composent l'application à déployer, afin d'en proposer une abstraction uniforme de contrôle et de supervision, répondant ainsi aux objectifs de généricité.

- un modèle de communication assurant les échanges entre ces entités de contrôle, au moyen de messages échangés au travers d'un bus décentralisé (i.e. communication en mode point à point), asynchrone (i.e. pas d'attente active dans l'envoi d'un message) et fiable (i.e. garantie de délivrance de chaque message envoyé).

Un configurateur a notamment pour fonction de :

- créer et configurer localement les composants dont il a la responsabilité, c'est-à-dire ceux associés à des éléments logiciels applicatifs déployés dans la machine virtuelle. Lors de cette première étape, chacun des configurateurs présents sur chacune des machines virtuelles applicatives agit indépendamment des autres ;

- interagir, dans un second temps, avec les configurateurs des autres machines virtuelles applicatives pour réaliser la configuration globale, c'est-à-dire l'établissement des liaisons entre composants distants, puis l'activation de l'application conformément à l'ordre de démarrage décrit par l'utilisateur.

Tel que représenté sur la figure 1, le gestionnaire de déploiement 210 s'exécutant dans une machine virtuelle 21 est dédié au déploiement de l'application APP ; le configurateur 220 s'exécutant dans une machine virtuelle 22 prend en charge le déploiement du composant 230 de l'application APP. Le composant 230 comprend un élément logiciel 240 localisé sur la machine virtuelle 22 et s'exécutant dans le plan applicatif. Un configurateur et le composant dont il est responsable sont co-localisés sur la même machine virtuelle.

Aucune limitation n'est attachée à cette représentation. En effet, le configurateur 220 peut également prendre en charge le déploiement d'autres composants de l'application APP. Leur nombre varie pour chaque application APP. D'autres configurateurs responsables de composants de l'application APP peuvent également être déployés sur d'autres machines virtuelles. Le gestionnaire de déploiement 210 et le configurateur 220 peuvent également être co-localisés sur une même machine virtuelle. Tels que représentées les machines virtuelles 21 et 22 sont co-localisées sur une machine physique 20. Aucune limitation n'est également attachée à cette représentation, les machines virtuelles pouvant être déployées sur des machines physiques distinctes.

Le gestionnaire de déploiement 210 et le ou les configurateurs 220 responsables de composants applicatifs forment un système de gestion d'une configuration 200 d'une application virtualisée.

Un composant est une entité d'administration de l'élément logiciel sous-jacent. Chaque composant expose :

- tout ou partie des propriétés de configuration de l'élément logiciel applicatif, accessibles en lecture et/ou en écriture ;
- des interfaces : une interface correspond à un point d'interconnexion entre composants. Une interface peut être cliente ou serveur, correspondant respectivement à la notion de service requis et de service fourni.
- une propriété de placement de l'élément logiciel au sein de l'infrastructure d'exécution de l'application (i. e. les machines virtuelles).

Les dépendances de configuration et d'activation entre éléments logiciels sont capturées au sein de cette architecture applicative à l'aide de liaisons. Une liaison permet d'interconnecter l'interface client d'un composant A à l'interface serveur d'un composant B, indiquant ainsi que A dépend de B d'un point de vue de sa configuration et/ou de son activation. Une liaison est dite obligatoire lorsque l'interface client du composant A considérée doit être liée à une interface serveur du composant B, au travers de cette liaison, pour que le composant A puisse être activé. Dans le cas contraire, c'est-à-dire quand le composant A peut être activé indépendamment du fait qu'il existe ou non une liaison entre l'interface client du composant A et une interface serveur du composant B, la liaison est optionnelle.

Dans le système VAMP, les composants Fractal sont implémentés à l'aide de « wrappers » (conteneurs), ou composants. Un « wrapper » est un objet Java (POJO pour « Plain Old Java Object ») qui implémente le comportement du composant au travers d'une interface incluant des opérations d'administration élémentaires :

- *export* : émission des attributs de configuration de l'élément logiciel encapsulé, relatives à une interface serveur du composant ;
- *bind* : configuration de l'élément logiciel encapsulé lors de la réception d'attributs émis par un autre composant, relativement à une interface client du composant ;
- *unbind* : destruction de la liaison entre l'interface client du composant et l'interface serveur d'un autre composant à laquelle elle était associée ;
- *start* / *stop* / *restart :* opérations de démarrage / arrêt / redémarrage de l'élément logiciel sous-jacent.

Un « wrapper » correspond ainsi à une instance réelle d'un composant. Par la suite, on utilise le terme composant.

Sur réception d'une requête de déploiement, le gestionnaire de déploiement 210 effectue les traitements suivants en fonction des informations contenues dans la requête :

- P1 : génération, si nécessaire, des images logicielles (i. e. piles logicielles) nécessaires à la mise en œuvre de l'application. Chaque image virtuelle est constituée d'un système d'exploitation, d'un ensemble d'intergiciels (ou « middleware ») ainsi que de binaires et de données applicatifs. Chaque image correspond au disque principal d'une (ou de plusieurs) des machines virtuelles qui composent l'infrastructure d'exécution de l'application à déployer. Outre ces éléments, le gestionnaire de déploiement insère dans chaque pile logicielle les binaires d'un configurateur ainsi que d'un ensemble de composants.
- P2 : publication des images virtuelles dans l'ensemble des plates-formes IaaS (« Infrastructure as a Service ») requises pour effectuer le déploiement de l'application. Il s'agit de les enregistrer dans le système de stockage de la plate-forme et de les rendre accessibles au travers de son répertoire d'images. Plus précisément, chaque image est publiée en fonction des besoins d'instanciation des machines virtuelles qui composent l'application.
- P3 : instanciation simultanée (i.e. en parallèle) de l'ensemble des machines virtuelles applicatives au sein du ou des plate(s)-forme(s) IaaS cible(s). Dès lors, chacune de ces machines virtuelles s'initialise (ou « boot ») indépendamment des autres, de façon asynchrone.

Lors de la phase de génération, un configurateur 220 est inséré dans une image virtuelle par le gestionnaire de déploiement 210 lors du traitement P1. Le configurateur 220 est ensuite activé automatiquement à l'issue du démarrage de la machine virtuelle 21.

Le configurateur 220 effectue alors, de manière indépendante vis-à-vis des autres configurateurs, les traitements suivants :

- A1- enregistrement dans le bus : le configurateur 220 contacte le gestionnaire de déploiement 210 pour indiquer qu'il souhaite s'enregistrer dans le bus à messages en tant que nouvel agent. Au terme de cette phase d'enregistrement, le gestionnaire de déploiement 210 transmet au configurateur le modèle de l'application à déployer.
- A2- instanciation des composants : à partir du modèle de l'application à déployer reçu, le configurateur 220 crée et configure localement les composants 230 dont il a la responsabilité et qui correspondent aux éléments logiciels 240 situés sur la machine virtuelle 21 où il se trouve.
- A3- résolution des contraintes de configuration distante : pour chaque composant dont il a la responsabilité, le configurateur 220 détermine à partir du modèle de l'application, s'il est nécessaire de transmettre des informations de configuration à destination de composants situés sur d'autres machines virtuelles applicatives. En d'autres termes, le configurateur 220 répertorie les composants locaux qui exposent une interface serveur et pour chacun d'eux, il émet un message RemoteBindingNotification contenant les informations de configuration associées vers le(s) configurateur(s) responsable du (ou des) composants dont une interface client est connectée à l'interface serveur considérée.
- A4- résolution des contraintes d'activation : une fois les dépendances de configuration résolues, le configurateur 220 active les composants qui peuvent l'être : il s'agit des composants dont l'ensemble des interfaces client qui participent à une liaison obligatoire sont reliées à l'interface serveur d'un composant déjà démarré. Lorsqu'un composant est démarré, le configurateur 220 envoie un message StartNotification à l'ensemble des configurateurs dont au moins un composant dépend du composant nouvellement activé.
- A5- réaction à la réception d'événements : une fois ces étapes effectuées, chaque configurateur 220 peut recevoir des messages issus d'autres configurateurs. Lorsqu'un configurateur reçoit un message RemoteBindingNotification, il le transmet au composant destinataire du message et celui-ci procède aux opérations de configuration relatives aux informations contenues dans le message. Lorsqu'un configurateur reçoit un message StartNotification, il détermine si le(s) composant(s) impacté(s) par ce démarrage peut (peuvent) à leur tour être démarré(s).

Pour chaque composant dont il a la charge, le configurateur 220 envoie au gestionnaire de déploiement 210 une notification de changement d'état, une fois le composant démarré.

A chaque réception d'une notification de changement d'état par le gestionnaire de déploiement 210, ce dernier détermine si le déploiement de l'application dans son ensemble est terminé.

Le procédé de déploiement initial est également fiable vis-à-vis des pannes franches de l'infrastructure d'exécution (i.e. machines virtuelles, réseau, ...). En effet, lors de la création d'une machine virtuelle, le gestionnaire de déploiement arme une temporisation correspondant à une durée maximale de démarrage d'une machine virtuelle. Si cette temporisation arrive à expiration, le

gestionnaire de déploiement demande de nouveau la création d'une nouvelle machine virtuelle tout en prenant soin de demander la destruction éventuelle de l'instance précédente (cas des faux positifs). Cette opération repose sur le fait que l'infrastructure de cloud est elle-même fiable. De plus, dès son instanciation, chaque configurateur initie un mécanisme de type « heart beat » par un envoi à intervalles réguliers d'un signal d'une preuve de vie à destination du gestionnaire de déploiement. Si ce signal, émis régulièrement, ne parvient pas dans un délai maximum au gestionnaire de déploiement, celui-ci procède au remplacement de la machine virtuelle défaillante par une instance de remplacement. Ce remplacement s'accompagne de l'élimination d'éventuels cas « faux positifs ». Tous les configurateurs qui participent au déploiement de l'application sont informés lors d'une panne d'une machine virtuelle et donc d'un configurateur. Dès lors, les configurateurs voisins, en termes de dépendances de configuration et d'activation, réémettent les messages qu'ils avaient déjà émis en direction de l'instance précédente. Symétriquement, à réception d'un message RemoteBindingNotification (respectivement d'un message StartNotification) de l'instance de remplacement, un composant procède à une opération de reconfiguration (respectivement de redémarrage).

Les figures 2a et 2b représentent schématiquement deux applications déployées sur une infrastructure IaaS.

La **figure 2a** correspond à une application comprenant trois composants C0, C1, C2 respectivement déployés sur des machines virtuelles VM0, VM1, VM2. Le composant C2 comprend une unique interface, qui est une interface serveur destinée à être reliée à une interface client du composant C1 par une liaison obligatoire. Cette liaison est notée L12. Le composant C1 comprend deux interfaces : une interface serveur destinée à être reliée à une interface client du composant C0 par une liaison obligatoire (notée L01) et l'interface client destinée à être reliée à l'interface serveur du composant C2 (liaison L12). Le composant C0 comprend une unique interface, qui est l'interface client destinée à être reliée à l'interface serveur du composant C1. Une fois enregistré sur le bus à messages, le configurateur responsable du composant C2 émet un message RemoteBindingNotification contenant les informations de configuration associées vers le configurateur responsable du composant C1 dont l'interface client est connectée à l'interface serveur qu'il propose. Le composant C2 ne disposant pas d'une interface client participant à une liaison obligatoire, le configurateur responsable du composant C2 démarre le composant C2 et envoie un message StartNotification au configurateur responsable du composant C1.

Le configurateur responsable du composant C1 émet un message RemoteBindingNotification contenant les informations de configuration associées vers le configurateur responsable du composant C0 dont l'interface client est connectée à l'interface serveur qu'il propose. Le composant C1 disposant d'une interface client participant à une liaison obligatoire, le configurateur responsable du composant C1 vérifie si le composant C2 a démarré. Si

tel est le cas, le configurateur responsable du composant C1 démarre le composant C1 et envoie un message StartNotification au configurateur responsable du composant C0.

Le configurateur responsable du composant C0, ce dernier disposant d'une interface client participant à une liaison obligatoire, vérifie si le composant C1 a démarré. Si tel est le cas, le configurateur responsable du composant C0 démarre le composant C0 et envoie un message StartNotification au configurateur responsable du composant C1.

On constate sur cet exemple que les composants démarrent dans l'ordre suivant : composant C2, composant C1, composant C0. En effet, par exemple si le composant C0 démarrait avant les deux autres composants, l'application ne pourrait s'exécuter et rendre le service attendu à un utilisateur.

La **figure 2b** correspond à une application plus complexe, comprenant trois composants C0, C1, C2 respectivement localisés sur trois machines virtuelles VM0, VM1, VM2. Chacun des composants C0, C1, C2 dispose de deux interfaces client et de deux interfaces serveur. La première interface client du composant C0 est reliée à une première interface serveur du composant C1 par une liaison L01. La deuxième interface client du composant C0 est reliée à une première interface serveur du composant C2 par une liaison L02. La première interface serveur du composant C0 est reliée à une première interface client du composant C1 par une liaison L10 obligatoire. La deuxième interface serveur du composant C0 est reliée à une première interface client du composant C2 par une liaison L20 obligatoire. La deuxième interface client du composant C1 est reliée à une deuxième interface serveur du composant C2 par une liaison L12. La deuxième interface serveur du composant C1 est reliée à une deuxième interface client du composant C2 par une liaison L21 obligatoire. Dans cet exemple, pour démarrer le composant C2, il est nécessaire que les composants C0 et C1 soient démarrés, et pour démarrer le composant C1, il est nécessaire que le composant C0 soit démarré. On constate sur cet exemple que les composants démarrent dans l'ordre suivant : composant C0, composant C1, composant C2.

Une fois le déploiement réalisé, le gestionnaire de déploiement 210 mémorise un modèle à l'exécution MOD_Ex (ou *model@runtime*), représentant l'état de déploiement courant de l'application. Ce modèle à l'exécution comprend le modèle de l'application déployée, ou modèle d'application courant, et des informations liées à l'instance de l'application telles que :

- un état d'avancement du déploiement des machines virtuelles applicatives ;
- un état de déploiement des composants applicatifs qui traduit l'avancement des opérations de configuration et d'activation des éléments logiciels sous-jacents ;
- un état d'avancement dans l'établissement des liaisons entre composants applicatifs ;
- un état de génération des piles logicielles (i.e. images logicielles) associées aux machines virtuelles applicatives ;

- un ensemble d'informations contextuelles tels que les adresses réseaux (adresses IP) des machines virtuelles applicatives, leur identifiant dans la plate-forme d'IaaS dans laquelle elles sont instanciées, etc.

Ce modèle à l'exécution MOD_Ex est construit à partir du modèle de l'application contenu dans la requête de déploiement émise par l'utilisateur du système VAMP ainsi que par les informations remontées au cours du déploiement par chaque configurateur. Le gestionnaire de déploiement 210 dispose ainsi d'une vue globale courante de l'avancement du déploiement de l'application.

Au niveau de chaque configurateur 220, ce modèle à l'exécution est mis à jour :

- soit par le configurateur lui-même, pour les éléments locaux à la machine virtuelle considérée ;
- soit par des mises à jour émises par le gestionnaire de déploiement 210.

La **figure 3** représente des étapes du procédé de reconfiguration mis en œuvre par un gestionnaire de déploiement 210.

Dans une étape E1, le gestionnaire de déploiement 210 reçoit une demande de modification du modèle à l'exécution vers un modèle cible MOD_T. Ce modèle cible correspond à un nouveau modèle à appliquer et décrit une nouvelle architecture de l'application APP. Cette demande de déploiement est reçue directement d'un utilisateur du système VAMP ou par l'intermédiaire du gestionnaire VAMP. Dans ce modèle cible, des éléments peuvent être supprimés, des éléments peuvent être arrêtés (par exemple en raison d'une opération de maintenance), des éléments peuvent être créés ou modifiés.

Dans une première phase φ1, dans une étape E2, le gestionnaire de déploiement 210 détermine un modèle intermédiaire MOD_Int. Plus précisément, le gestionnaire de déploiement 210 détermine par comparaison entre le modèle à l'exécution MOD_Ex et le modèle cible MOD_T les éléments à supprimer ou les éléments à arrêter. Le terme élément désigne aussi bien les machines virtuelles VM, les piles logicielles, les composants et les liaisons. Le gestionnaire de déploiement 210 mémorise les éléments du modèle à l'exécution qui ne sont pas à supprimer ou à arrêter dans le modèle intermédiaire MOD_Int. Ainsi, ce modèle intermédiaire ne contient ni les éléments à ajouter ni les modifications de configuration à effectuer sur des éléments existants tels que définis dans le modèle cible.

Dans une étape E3, le gestionnaire de déploiement 210 transmet aux configurateurs 220 le modèle intermédiaire MOD_Int pour déploiement.

Dans une étape E4, le gestionnaire de déploiement 210 attend une confirmation de chacun des configurateurs que le modèle intermédiaire MOD_Int a bien été appliqué. Plus précisément, la confirmation correspond à une réception d'une notification de changement d'état envoyée par le configurateur 220 pour chaque composant concerné. Une fois toutes les notifications de changement d'état reçues, les éléments qui étaient à supprimer ne sont plus présents sur les

machines virtuelles, les éléments qui étaient à arrêter l'ont été et les éléments impactés par la suppression ou l'arrêt d'un autre élément ont été arrêtés. La première phase φ1 est alors terminée, les éléments absents du modèle d'application cible ayant été supprimés du modèle à l'exécution et les éléments arrêtés dans le modèle d'application cible ayant été arrêtés dans le modèle à l'exécution. Le modèle intermédiaire devient le nouveau modèle à l'exécution.

Dans une deuxième phase φ2, dans une étape E5, le gestionnaire de déploiement 210 procède alors par comparaison entre le modèle à l'exécution MOD_Ex et le modèle cible MOD_T, à la génération, lorsqu'un composant nécessite la mise en œuvre d'un élément logiciel non publié dans une image logicielle, des nouvelles images logicielles (i. e. piles logicielles) nécessaires à la mise en œuvre de l'application (traitement P1 décrit précédemment), à la publication de ces nouvelles images virtuelles (traitement P2 décrit précédemment), et à la reconfiguration des propriétés relatives à des machines virtuelles existantes (modification d'une machine virtuelle) et à l'instanciation de l'ensemble des nouvelles machines virtuelles applicatives (traitement P3 décrit précédemment).

Dans une étape E6, le gestionnaire de déploiement 210 transmet alors le modèle cible aux configurateurs 220, afin que ces derniers effectuent les opérations d'ajout, d'activation et de modification sur les composants applicatifs.

Dans une étape E7, le gestionnaire de déploiement 210 attend une confirmation de chacun des configurateurs que le modèle cible MOD_T a bien été appliqué. Plus précisément, la confirmation correspond à une réception d'une notification de changement d'état envoyée par le configurateur 220 pour chaque composant concerné. Une fois toutes les notifications de changement d'état reçues, les éléments à ajouter ont été créés sur les machines virtuelles, les éléments à activer l'ont été et les éléments à modifier ont été mis à jour. Ceci termine la deuxième phase φ2.

Ainsi, par la mise en œuvre de ces deux phases, la reconfiguration du modèle à l'exécution vers le modèle cible MOD_T a été effectuée sans nécessiter un nouveau déploiement complet de l'application sur les machines virtuelles et sans créer d'incohérences d'état entre les différents composants. La cohérence architecturale de l'application a été maintenue.

La **figure 4a** illustre des étapes d'un procédé de reconfiguration d'un modèle à l'exécution MOD_Ex vers un modèle à appliquer mises en œuvre par un configurateur 220 de l'application APP.

Dans une étape F1, le configurateur 220 reçoit un modèle à appliquer.

Dans une étape F2, le configurateur 220 détermine en fonction du modèle à l'exécution MOD_Ex une ou des actions à effectuer pour obtenir le modèle à appliquer reçu. Ces actions correspondent le cas échéant à une suppression de composant, un arrêt de composant, une activation de composant, une modification de composant ou un ajout de composant. Un composant

doit être arrêté lorsqu'il dépend en tant que client d'un composant à supprimer ou à arrêter au travers d'une liaison obligatoire. Une interface client d'un composant relié par une liaison obligatoire à une interface serveur d'un autre composant doit être déconfigurée lorsque cet autre composant est à supprimer et ceci doit être effectué préalablement à la suppression de cet autre composant. Une suppression d'un composant correspond notamment à une désinstallation du logiciel correspondant, à une libération des ressources et à un nettoyage des données.

Dans une étape F3, le configurateur 220 exécute le cas échéant l'action ou les actions déterminées à l'étape F2. Au cours de l'exécution de cette étape F3, le configurateur 220 envoie au gestionnaire de déploiement 210 une notification de changement d'état pour chaque composant concerné.

Ces étapes F1 à F3 sont mises en œuvre lors d'une reconfiguration d'un modèle à l'exécution vers un modèle cible, une première fois avec le modèle intermédiaire MOD_Int déterminé par le gestionnaire de déploiement à l'étape E2 et une deuxième fois avec le modèle cible MOD_T transmis par le gestionnaire de déploiement à l'étape E6.

La **figure 4b** représente plus précisément l'étape F3 mise en œuvre par le configurateur 220.

Dans une étape F30, le configurateur 220 vérifie s'il a une ou plusieurs actions à effectuer.

S'il n'a pas d'action à effectuer, l'étape F3 est terminée.

Si le configurateur 220 a au moins une action à effectuer, dans une étape F31, le configurateur 220 détermine si l'action appartient à un groupe comprenant une suppression ou un arrêt d'un composant 230. Si tel est le cas, dans une étape F32, le configurateur 220 vérifie s'il existe une interface serveur du composant 230 reliée par une liaison obligatoire à une interface client d'un autre composant. Si tel est le cas, le composant 230 ne pouvant pas être arrêté immédiatement, dans une étape F320, le configurateur 220 se place alors en attente de réception d'un message d'arrêt de notification StopNotification en provenance du configurateur responsable de cet autre composant. Un tel message est reçu dans une étape F321. Toujours dans cette étape F321, le configurateur 220 met à jour sa représentation locale du modèle à l'exécution en fonction de ce message reçu. Le configurateur 220 met de nouveau en œuvre l'étape F32 de vérification s'il existe une interface serveur du composant 230 reliée par une liaison obligatoire à une interface client d'un autre composant.

S'il est vérifié à l'étape F32 qu'il n'existe pas d'interface serveur du composant 230 reliée par une liaison obligatoire à une interface client d'un autre composant, le composant 230 est arrêté car il ne joue pas un rôle de serveur vis-à-vis d'un des autres composants de l'application. Dans une étape F322, le configurateur 220 envoie un message d'arrêt de notification StopNotification pour une interface client reliée par une liaison à une interface serveur d'un autre

composant s'il en existe. Plus précisément, ce message d'arrêt de notification est envoyé aux configurateurs responsables de ces autres composants.

Dans une étape F323, le configurateur 220 exécute l'action à effectuer. Lorsque l'action est un arrêt du composant, le configurateur 220 arrête l'exécution du composant 230 et modifie un état du composant à « stopped ». Lorsque l'action est une suppression du composant, le configurateur 220 arrête l'exécution du composant 230, relâche les ressources utilisées par ce composant et modifie un état du composant à « deleted ».

Dans une étape F324, le configurateur 220 mémorise dans sa représentation locale du modèle à l'exécution le nouvel état, « stopped » ou « deleted », du composant 230.

Dans une étape F325, le configurateur 220 envoie au gestionnaire de déploiement 210 une notification de changement d'état, comprenant le nouvel état du composant 230.

Le configurateur 220 exécute de nouveau l'étape F30, afin de vérifier s'il a encore une ou plusieurs actions à effectuer.

S'il est vérifié à l'étape F31 que l'action ne correspond pas à une suppression ou un arrêt d'un composant 230, cette action appartient alors à un groupe comprenant une création, une activation ou une modification du composant 230.

Dans une étape F311, le configurateur 220 exécute le traitement A2 décrit précédemment. Plus précisément, s'il s'agit d'une création d'un composant, le configurateur 220 crée le composant correspondant. Dans les deux cas, création et modification, le configurateur 220 configure localement le composant correspondant.

Dans une étape F312, le configurateur 220 exécute le traitement A3 décrit précédemment. Plus précisément, le configurateur 220 répertorie les composants locaux qui exposent une interface serveur et pour chacun d'eux, il émet un message RemoteBindingNotification contenant les informations de configuration associées vers le(s) configurateur(s) responsable du (ou des) composant(s) dont une interface client est connectée à l'interface serveur considérée. S'il s'agit uniquement d'une modification d'une propriété de configuration d'un composant, il n'y a pas d'émission de message RemoteBindingNotification.

Dans une étape F313, le configurateur 220 exécute le traitement A4 décrit précédemment. Plus précisément, le configurateur 220 active le composant dont l'ensemble des interfaces client qui participent à une liaison obligatoire sont reliées à l'interface serveur d'un composant déjà démarré. Lorsqu'un composant est démarré, le configurateur 220 envoie un message StartNotification à l'ensemble des configurateurs dont au moins un composant dépend du composant nouvellement activé ou modifié.

Dans une étape F314, le configurateur 220 modifie l'état du composant à « started ».

Dans une étape F315, le configurateur 220 mémorise dans sa représentation locale du modèle à l'exécution le nouvel état, « started », du composant 230.

Le configurateur 220 exécute alors l'étape F325, précédemment décrite, d'envoi au gestionnaire de déploiement 210 d'une notification du nouvel état du composant 230.

On comprend de cette description de l'étape F3, que lors de la réception d'un modèle intermédiaire, les étapes F32x sont notamment mises en œuvre pour l'arrêt ou la suppression d'un composant et lors de la réception du modèle cible, les étapes F31x sont notamment mises en œuvre pour la création, l'activation ou la modification d'un composant.

De retour à l'exemple de la figure 2a, on se place dans le cas où le composant C1 n'est plus présent dans le modèle cible. Le composant C0 doit être arrêté car une de ses interfaces client est reliée par une liaison obligatoire L01 à une interface serveur du composant C1. Le composant C1 doit être supprimé. Le composant C2 n'est pas modifié car il n'a pas d'interface client reliée par une liaison obligatoire à une interface serveur du composant C1.

La description qui suit est effectuée de manière séquentielle. Il est ici souligné qu'il s'agit d'une question de lisibilité et que les différents configurateurs travaillent en parallèle et de manière asynchrone.

Le configurateur du composant C0 détermine que ce composant doit être arrêté. Il vérifie (étape F32) qu'il n'a pas d'interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant. Il envoie (étape F322) un message d'arrêt de notification StopNotification au configurateur responsable du composant C1. Puis il arrête le composant C0 (étape F323).

Le configurateur du composant C1 détermine que ce composant doit être supprimé. Il vérifie (étape F32) qu'il a une interface serveur reliée par une liaison obligatoire à une interface client du composant C0. Tant que le composant C0 est actif, il se place en attente (étape F320). Une fois le message d'arrêt de notification StopNotification reçu (étape F321) en provenance du composant C0, comme il n'a plus d'interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant (étape F32), il envoie (étape F322) un message d'arrêt de notification StopNotification au configurateur responsable du composant C2. Puis il arrête et supprime le composant C1 (étape F323).

Le configurateur du composant C2 détermine qu'il n'a pas d'action à effectuer. Sur réception du message d'arrêt de notification StopNotification en provenance du composant C1, il met à jour sa représentation locale du modèle à l'exécution. On constate ainsi qu'il n'est pas possible de supprimer le composant C1 tant que le composant C0 n'est pas arrêté.

De retour à l'exemple de la figure 2b, on se place dans le cas où dans le modèle cible le composant C1 doit être supprimé et remplacé par un composant C4. Ce composant C4 s'interface avec les composants C0 et C2 par des liaisons similaires à celles du composant C1 avec les composants C0 et C2. Le gestionnaire de déploiement 210 détermine (étape E2) un modèle intermédiaire, dans lequel le composant C1 n'est plus présent. Le composant C0 n'est pas modifié car aucune de ses interfaces client n'est reliée par une liaison obligatoire à une interface serveur du

composant C1 (la liaison L01 est optionnelle). Le composant C1 doit être supprimé. Le composant C2 est à arrêter car il a une interface client reliée par une liaison obligatoire L21 à une interface serveur du composant C1.

Le configurateur du composant C0 détermine qu'il n'a pas d'action à effectuer.

Le configurateur du composant C1 détermine que ce composant doit être supprimé. Il vérifie (étape F32) qu'il a une interface serveur reliée par une liaison obligatoire L21 à une interface client du composant C2 (L01 est une liaison optionnelle). Tant que le composant C2 est actif, il se place en attente (étape F320). Une fois le message d'arrêt de notification StopNotification reçu (étape F321) en provenance du composant C2, comme il n'a plus d'interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant (étape F32), il envoie (étape F322) un message d'arrêt de notification StopNotification aux configurateurs responsables des composants C0 et C2. Puis il arrête et supprime le composant C1 (étape F323).

Le configurateur du composant C2 détermine que ce composant doit être arrêté. Il vérifie (étape F32) qu'il n'a pas d'interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant (L02 et L12 sont des liaisons optionnelles). Il envoie (étape F322) un message d'arrêt de notification StopNotification aux configurateurs responsables des composants C0 et C1. Puis il arrête (étape F323) le composant C2.

Sur réception du message d'arrêt de notification StopNotification en provenance du composant C2, les configurateurs responsables des composants C0 et C1 mettent à jour leur représentation locale du modèle à l'exécution. Sur réception du message d'arrêt de notification StopNotification en provenance du composant C1, les configurateurs responsables des composants C0 et C2 mettent à jour leur représentation locale du modèle à l'exécution. On constate ainsi qu'il n'est pas possible de supprimer le composant C1 tant que le composant C2 n'est pas arrêté.

Le gestionnaire de déploiement 210 reçoit (étape E4) les notifications de changement d'état en provenance des configurateurs responsables des composants C1 (état « deleted ») et C2 (état « stopped »). La première phase $\varphi 1$ est ainsi terminée.

Le gestionnaire de déploiement 210 procède (étape E5), par comparaison entre le modèle à l'exécution et le modèle cible, à la génération de la nouvelle image logicielle du composant C4, à la publication de cette nouvelle image virtuelle et à la mise à jour des propriétés relatives aux machines virtuelles existantes et à l'instanciation de la nouvelle machine virtuelle. Le gestionnaire de déploiement 210 transmet (étape E6) le modèle cible aux configurateurs 220, afin que ces derniers effectuent les opérations nécessaires sur les composants applicatifs.

Le configurateur 220 est par exemple en charge de la création du composant C4.

Le configurateur 220 crée et configure (étape F311) localement le composant C4 correspondant. Lorsque le composant C4 expose une interface serveur, le configurateur 220 émet (étape F312) un message RemoteBindingNotification contenant les informations de configuration

associées vers le(s) configurateur(s) responsable du (ou des) composant dont une interface client est connectée à l'interface serveur considérée.

Le configurateur 220 active le composant dont l'ensemble des interfaces client qui participent à une liaison obligatoire sont reliées à l'interface serveur d'un composant déjà démarré.

Le composant C0 est déjà démarré.

Le composant C2 doit être redémarré. Le configurateur du composant C2 envoie un message StartNotification aux composants C0 et C4. L'état du composant C2 devient « started ».

Lorsque le composant C4 est démarré, le configurateur 220 envoie un message StartNotification aux composants C0 et C2. L'état du composant C4 devient « started ».

Le gestionnaire de déploiement 210 reçoit (étape E4) les notifications de changement d'état en provenance des configurateurs responsables des composants C4 (état « started ») et C2 (état « started »). Cette deuxième phase φ2 est ainsi terminée. Le modèle cible a été déployé et l'application est de nouveau opérationnelle.

Le procédé de reconfiguration en deux phases est asynchrone, réparti et fiable. Il s'appuie sur les éléments suivants :

- la définition et la mémorisation d'une représentation courante de l'application déployée (i.e. le modèle à l'exécution). Il permet au gestionnaire de déploiement de procéder à des comparaisons entre le modèle d'application cible à déployer et un état courant de l'application et de calculer les actions à effectuer en termes de reconfiguration.
- la comparaison entre deux versions du modèle d'application qui permet de déterminer les éléments (i.e. machines virtuelles, images, composants et liaisons) à modifier, à supprimer, à arrêter ou à ajouter.
- l'arrêt d'un composant applicatif, qui procède de façon symétrique à son activation en s'appuyant sur la définition des contraintes d'activation au moyen des liaisons.

Dans un mode de réalisation particulier, un estampillage est représentatif d'une version du modèle d'application à déployer. Cet estampillage est associé au modèle envoyé aux configurateurs. Les messages envoyés par les configurateurs incluent cet estampillage, afin de faire référence à la version du modèle d'application. Les estampilles sont ordonnées, de façon absolue, dans le temps. Il est donc possible, à partir de ces estampilles, de classer les versions du modèle d'application. Cette estampille peut être par exemple temporelle. Le modèle à l'exécution au sein du gestionnaire de déploiement ainsi qu'au niveau de chaque configurateur dispose de la valeur d'estampille qui lui est associé.

Lorsqu'il émet un message de type RemoteBindingNotification, StartNotification ou StopNotification, un configurateur enrichit le message de la valeur d'estampille dont il a connaissance.

De manière symétrique, lorsqu'un configurateur reçoit un message de type RemoteBindingNotification, StartNotification ou StopNotification, il procède de la façon suivante :

- si l'estampille contenue dans le message reçu est antérieure à l'estampille courante du configurateur (c'est-à-dire celle associée au dernier modèle qu'il a reçu), il n'effectue aucune opération et détruit le message, ce dernier étant considéré comme périmé.
- si l'estampille contenue dans le message est postérieure à l'estampille courante du configurateur, le message est mis en attente, jusqu'à ce que l'estampille du configurateur soit égale (suite à la réception d'une mise à jour du modèle) à l'estampille du message reçu. Le message est alors traité.
- si l'estampille du message et celle du configurateur sont égales, le message est traité immédiatement.

Ce mécanisme d'estampille permet de prendre en compte l'asynchronisme dans l'exécution des différents configurateurs. Il peut être vu comme un mécanisme de synchronisation deux à deux entre configurateurs.

L'estampille d'un configurateur est mise à jour lorsqu'il reçoit du gestionnaire de déploiement une nouvelle version du modèle d'application à déployer.

L'estampille du gestionnaire de déploiement est incrémentée à chaque nouvelle opération de déploiement.

Cet estampillage permet de garantir une synchronisation entre les différents composants.

La **figure 5** représente un dispositif mettant en œuvre un gestionnaire de déploiement dans un mode particulier de réalisation. Le gestionnaire de déploiement comprend notamment :

- un processeur 211 pour exécuter des instructions de code de modules logiciels ;
- une zone mémoire 212, agencée pour mémoriser une application qui comprend des instructions de code pour mettre en œuvre les étapes du procédé de reconfiguration ;
- une mémoire de stockage, non représentée, agencée pour stocker des données utilisées lors de la mise en œuvre du procédé de reconfiguration ;
- un module 213 d'obtention d'un nouveau modèle d'application, dit modèle d'application cible ;
- un module 214 de détermination d'un modèle intermédiaire, dans lequel un composant présent dans le modèle à l'exécution et absent du modèle d'application cible est supprimé et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter ;
- un module 215 d'envoi à un configurateur responsable d'un composant applicatif, d'une commande de déploiement d'un modèle à appliquer ;
- un module 216 de génération d'au moins une nouvelle image logicielle nécessaire à la mise en œuvre d'un nouveau composant, et d'instanciation sous forme d'une machine virtuelle de l'infrastructure ;

- un module 217 de commande, agencé pour commander le module d'envoi pour envoyer le modèle intermédiaire déterminé et lorsque le modèle intermédiaire est déployé sur l'infrastructure, pour activer le module de génération/instanciation et pour commander le module d'envoi pour envoyer le modèle d'application cible.

Dans un mode de réalisation particulier, le module 216 de génération est activé lorsqu'un composant nécessite la mise en œuvre d'un élément logiciel non publié dans une image logicielle.

La **figure 6** représente un dispositif mettant en œuvre un configurateur dans un mode particulier de réalisation. Le configurateur comprend notamment :

- un processeur 221 pour exécuter des instructions de code de modules logiciels ;
- une zone mémoire 222, agencée pour mémoriser une application qui comprend des instructions de code pour mettre en œuvre les étapes du procédé de reconfiguration ;
- une mémoire de stockage, non représentée, agencée pour stocker des données utilisées lors de la mise en œuvre du procédé de reconfiguration ;
- un module de réception 223 en provenance d'un gestionnaire de déploiement, d'une commande de déploiement d'un modèle à appliquer, ce modèle correspondant au modèle cible ou à un modèle intermédiaire dans lequel un composant présent dans le modèle à l'exécution et absent du modèle d'application cible est supprimé et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter ;
- un module d'exécution 224 d'une action, ladite action appartenant à un groupe comprenant une suppression d'un composant lorsque ledit composant est absent du modèle à appliquer, un arrêt d'un composant lorsque ledit composant est indiqué à arrêter dans le modèle à appliquer, un arrêt d'un composant lorsque ledit composant comprend une interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant absent du modèle à appliquer, une création d'un nouveau composant, une modification d'un composant, une activation d'un composant arrêté lorsque ledit composant comprend une interface serveur reliée par une liaison obligatoire à une interface client d'un nouveau composant ;
- un module d'envoi 225 au gestionnaire de déploiement d'une notification de changement d'état du composant.

On comprend que les processeurs et les mémoires sont des ressources matérielles qui appartiennent à une machine physique. Ces ressources sont destinées à être virtualisées par l'hyperviseur et mises à disposition des machines virtuelles.

L'invention vise également un programme d'ordinateur sur un ou plusieurs support(s) d'enregistrement, ce programme étant susceptible d'être mis en œuvre dans un système de gestion de configuration ou plus généralement dans un ordinateur, ce programme comportant des instructions adaptées à la mise en œuvre des étapes d'un procédé de reconfiguration tel que décrit précédemment.

Ce programme peut utiliser n'importe quel langage de programmation, et être sous la forme de code source, code objet, ou de code intermédiaire entre code source et code objet, tel que dans une forme partiellement compilée, ou dans n'importe quelle autre forme souhaitable.

Dans un mode de réalisation particulier, le gestionnaire de déploiement 210 est agencé pour mettre en œuvre le procédé de reconfiguration précédemment décrit. Il s'agit de préférence d'un module logiciel comprenant des instructions logicielles pour faire exécuter celles des étapes du procédé de reconfiguration précédemment décrit, mises en œuvre par un ordinateur. L'invention concerne donc aussi :

- un programme pour un gestionnaire de déploiement, comprenant des instructions de code de programme destinées à commander l'exécution des étapes du procédé de reconfiguration précédemment décrit, lorsque ledit programme est exécuté par cet ordinateur ;
- un support d'enregistrement lisible par un ordinateur sur lequel est enregistré le programme pour un gestionnaire de déploiement.

Dans un mode de réalisation particulier, le configurateur 220 est agencé pour mettre en œuvre le procédé de reconfiguration précédemment décrit. Il s'agit de préférence d'un module logiciel comprenant des instructions logicielles pour faire exécuter celles des étapes du procédé de reconfiguration précédemment décrit, mises en œuvre par un ordinateur. L'invention concerne donc aussi :

- un programme pour un configurateur, comprenant des instructions de code de programme destinées à commander l'exécution des étapes du procédé de reconfiguration précédemment décrit, lorsque ledit programme est exécuté par cet ordinateur ;
- un support d'enregistrement lisible par un ordinateur sur lequel est enregistré le programme pour un configurateur.

Les modules logiciels peuvent être stockés dans ou transmis par un support d'enregistrement. Le support d'enregistrement peut être n'importe quelle entité ou dispositif capable de stocker le programme. Par exemple, le support peut comporter un moyen de stockage, tel qu'une ROM, par exemple un CD ROM ou une ROM de circuit microélectronique, ou encore un moyen d'enregistrement magnétique, par exemple une disquette (floppy disc) ou un disque dur. D'autre part, le support d'enregistrement peut être un support transmissible tel qu'un signal électrique ou optique, qui peut être acheminé via un câble électrique ou optique, par radio ou par d'autres moyens. Le programme peut être en particulier téléchargé sur un réseau de type Internet. Alternativement, le support d'enregistrement peut être un circuit intégré dans lequel le programme est incorporé, le circuit étant adapté pour exécuter ou pour être utilisé dans l'exécution du procédé en question.

La technique de reconfiguration a été décrite pour une mise en œuvre dans une plateforme conforme à VAMP. Cette technique peut également être transposée à d'autres

environnements virtuels, dans lesquels une application logicielle est représentée sous la forme d'un modèle d'application modélisant des dépendances entre entités du modèle, comprenant des composants applicatifs et des contraintes de déploiement, et le déploiement s'effectue de manière décentralisée.

5

# REVENDICATIONS

1. Procédé de reconfiguration d'un modèle d'application pour un déploiement d'une application logicielle, un modèle d'application comprenant au moins un composant applicatif et des contraintes de déploiement, un modèle d'application courant, dit modèle à l'exécution, étant déployé sur une infrastructure de machines virtuelles (21, 22) reposant sur au moins une machine physique (20), le procédé comprenant :

- obtention (E1) par un gestionnaire de déploiement (210) d'un nouveau modèle d'application, dit modèle d'application cible ;
- détermination (E2) par le gestionnaire de déploiement d'un modèle intermédiaire dans lequel, un composant présent dans le modèle à l'exécution et absent du modèle d'application cible est supprimé, et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter ;
- envoi (E3), par le gestionnaire de déploiement à un configurateur (220) responsable d'un composant applicatif (230), d'une commande de déploiement d'un modèle intermédiaire, comprenant ledit au moins un composant du modèle à l'exécution ;
- lorsqu'une notification de changement d'état a été reçue (E4) en provenance du configurateur pour chaque composant concerné par le déploiement du modèle intermédiaire sur l'infrastructure, envoi (E6) par le gestionnaire de déploiement à un configurateur responsable d'un composant applicatif, d'une commande de déploiement du modèle d'application cible,

le modèle d'application cible devenant le modèle à l'exécution une fois déployé sur l'infrastructure.

2. Procédé de reconfiguration selon la revendication 1, comprenant préalablement à l'envoi du modèle d'application cible, lorsqu'un composant nécessite la mise en œuvre d'un élément logiciel non publié dans une image logicielle, une génération (E5) d'au moins une nouvelle image logicielle nécessaire à la mise en œuvre du nouveau composant, publication et instanciation (E5) sous forme d'une machine virtuelle de l'infrastructure.

3. Procédé de reconfiguration selon la revendication 1, dans lequel une estampille ordonnée est associée à un modèle d'application envoyé à un configurateur.

4. Procédé de reconfiguration d'un modèle d'application pour un déploiement d'une application logicielle, un modèle d'application comprenant au moins un composant applicatif et des contraintes de déploiement, un modèle d'application courant, dit modèle à 'exécution, étant déployé sur une infrastructure de machines virtuelles (21, 22) reposant sur au moins une machine physique (20), le procédé comprenant :

- réception (F1), par un configurateur (220) responsable d'un composant applicatif (230) en provenance d'un gestionnaire de déploiement (210), d'une commande de déploiement d'un modèle intermédiaire, comprenant ledit au moins un composant du modèle à l'exécution et dans lequel un composant présent dans le modèle à l'exécution et absent du modèle d'application cible est supprimé et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter ;
- exécution (F3, F323) d'une action, ladite action appartenant à un groupe comprenant une suppression d'un composant lorsque ledit composant est absent du modèle intermédiaire, un arrêt d'un composant lorsque ledit composant est indiqué à arrêter, un arrêt du composant lorsque ledit composant comprend une interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant absent du modèle intermédiaire ;
- envoi (F3, F325) au gestionnaire de déploiement d'une notification de changement d'état du composant ;
- réception (F1), par un configurateur responsable d'un composant applicatif en provenance d'un gestionnaire de déploiement, d'une commande de déploiement du modèle d'application cible ;
- exécution (F3, F314) d'une action, ladite action appartenant à un groupe comprenant une création d'un nouveau composant, une modification d'un composant, une activation d'un composant arrêté lorsque ledit composant comprend une interface serveur reliée par une liaison obligatoire à une interface client d'un nouveau composant ;
- envoi (F3, F325) au gestionnaire de déploiement d'une notification de changement d'état du composant.

5. Procédé de reconfiguration selon la revendication 4, dans lequel un message envoyé à un configurateur responsable d'un composant lié comprend une estampille reçue avec un modèle d'application.

6. Procédé de reconfiguration selon la revendication 4, dans lequel une action de suppression ou d'arrêt d'un composant applicatif est effectuée lorsque ledit composant n'a pas d'interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant applicatif.

7. Procédé de reconfiguration selon la revendication 4, dans lequel lors de l'exécution de l'action de suppression ou d'arrêt d'un composant applicatif, le configurateur responsable dudit composant envoie (F322) un message d'arrêt de notification pour une interface client reliée par une liaison à une interface serveur d'un autre composant.

8. Dispositif (20) mettant en œuvre un gestionnaire de déploiement (210) s'exécutant sur une machine virtuelle (21) hébergée par le dispositif (20), ledit gestionnaire de déploiement comprenant pour reconfigurer un modèle d'application pour un déploiement d'une application logicielle, un modèle d'application comprenant au moins un composant applicatif et des contraintes de déploiement, un modèle d'application courant, dit modèle à l'exécution, étant déployé sur une infrastructure de machines virtuelles (21, 22) reposant sur au moins une machine physique :

- un module (213) d'obtention d'un nouveau modèle d'application, dit modèle d'application cible ;
- un module (214) de détermination d'un modèle intermédiaire, comprenant ledit au moins un composant du modèle à l'exécution et dans lequel un composant présent dans le modèle à l'exécution et absent du modèle d'application cible est supprimé et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter ;
- un module (215) d'envoi à un configurateur (220) responsable d'un composant applicatif (230), d'une commande de déploiement d'un modèle à appliquer ;
- un module (216) de génération d'au moins une nouvelle image logicielle nécessaire à la mise en œuvre d'un nouveau composant, et d'instanciation sous forme d'une machine virtuelle de l'infrastructure ;
- un module (217) de commande, agencé pour commander le module d'envoi pour envoyer le modèle intermédiaire déterminé et lorsqu'une notification de changement d'état a été reçue en provenance du configurateur pour chaque composant concerné par le déploiement du modèle intermédiaire sur l'infrastructure, pour activer le module de génération/instanciation et pour commander le module d'envoi pour envoyer le modèle d'application cible.

9. Dispositif selon la revendication 8, dans lequel le module de génération est activé pour générer une nouvelle image logicielle lorsqu'un composant nécessite la mise en œuvre d'un élément logiciel non publié dans une image logicielle.

10. Dispositif (20) mettant en œuvre un configurateur (220) s'exécutant sur une machine virtuelle (22) hébergée par le dispositif (20), ledit configurateur comprenant pour reconfigurer un modèle d'application pour un déploiement d'une application logicielle, un modèle d'application comprenant au moins un composant applicatif et des contraintes de déploiement, un modèle d'application courant, dit modèle à l'exécution, étant déployé sur une infrastructure de machines virtuelles (21, 22) reposant sur au moins une machine physique :

- un module de réception (223) en provenance d'un gestionnaire de déploiement (210), d'une commande de déploiement d'un modèle à appliquer, ce modèle correspondant au modèle cible ou à un modèle intermédiaire comprenant ledit au moins un composant du modèle à l'exécution et dans lequel, un composant présent dans le modèle à l'exécution et absent du modèle d'application cible

est supprimé, et un composant présent dans le modèle à l'exécution et arrêté dans le modèle d'application cible est indiqué à arrêter ;

- un module d'exécution (224) d'une action, ladite action appartenant à un groupe comprenant une suppression d'un composant lorsque ledit composant est absent du modèle à appliquer, un arrêt d'un composant lorsque ledit composant est indiqué à arrêter, un arrêt d'un composant lorsque ledit composant comprend une interface serveur reliée par une liaison obligatoire à une interface client d'un autre composant absent du modèle à appliquer, une création d'un nouveau composant, une modification d'un composant, une activation d'un composant arrêté lorsque ledit composant comprend une interface serveur reliée par une liaison obligatoire à une interface client d'un nouveau composant ;

- un module d'envoi (225) au gestionnaire de déploiement d'une notification de changement d'état du composant.

11 Programme pour un dispositif, comprenant des instructions de code de programme destinées à commander l'exécution des étapes du procédé reconfiguration selon l'une des revendications 1 à 3 mises en œuvre par le dispositif, lorsque ledit programme est exécuté par ledit dispositif.

12. Support d'enregistrement lisible par un dispositif sur lequel est enregistré le programme selon la revendication 11.

13. Programme pour un dispositif, comprenant des instructions de code de programme destinées à commander l'exécution des étapes du procédé reconfiguration selon l'une des revendications 4 à 7 mises en œuvre par le dispositif, lorsque ledit programme est exécuté par ledit dispositif.

14. Support d'enregistrement lisible par un dispositif sur lequel est enregistré le programme selon la revendication 13.

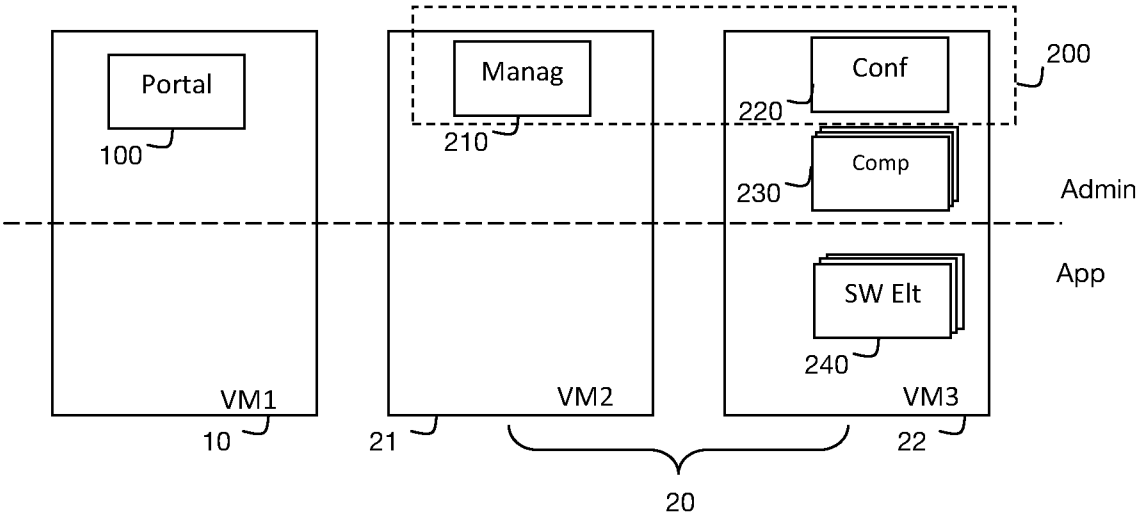



Fig. 1

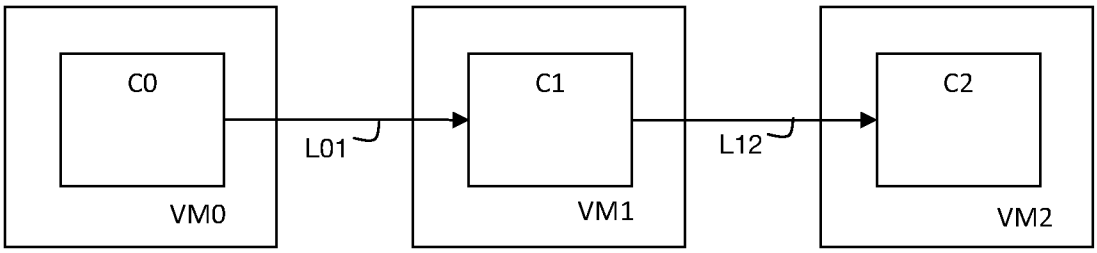



Fig. 2a

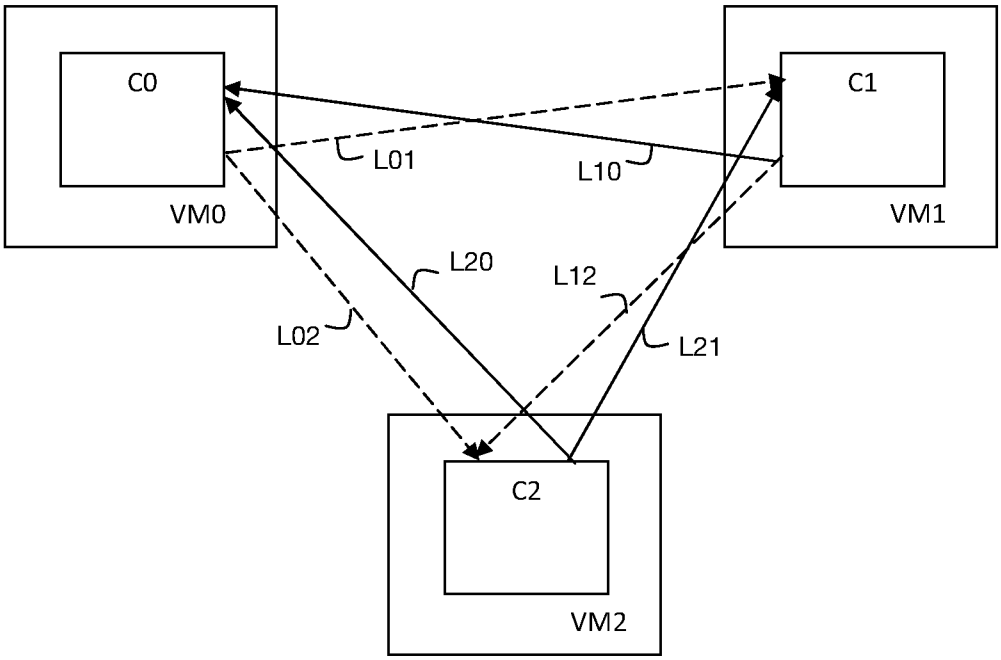



Fig. 2b

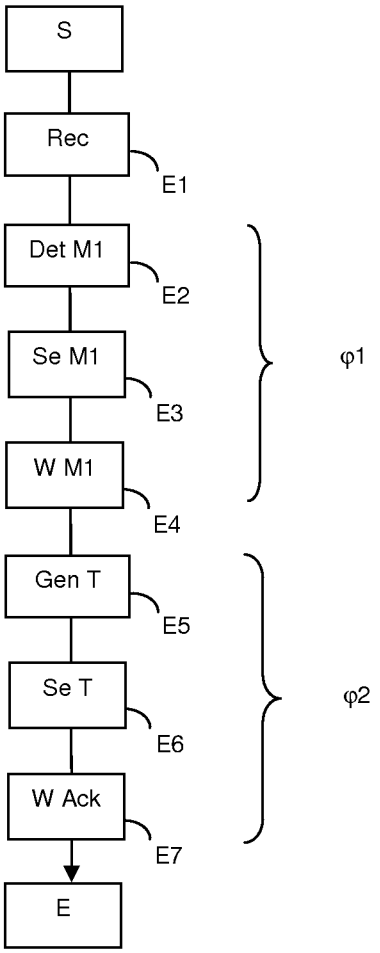



Fig. 3

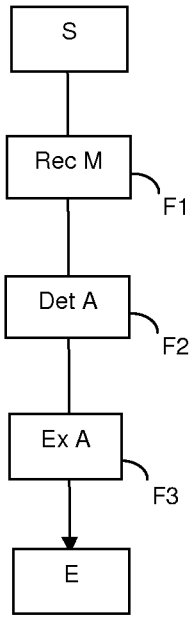



Fig. 4a

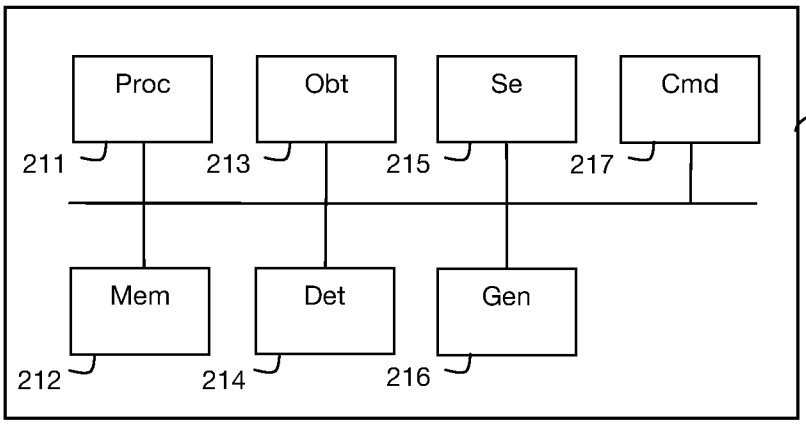



Fig. 5

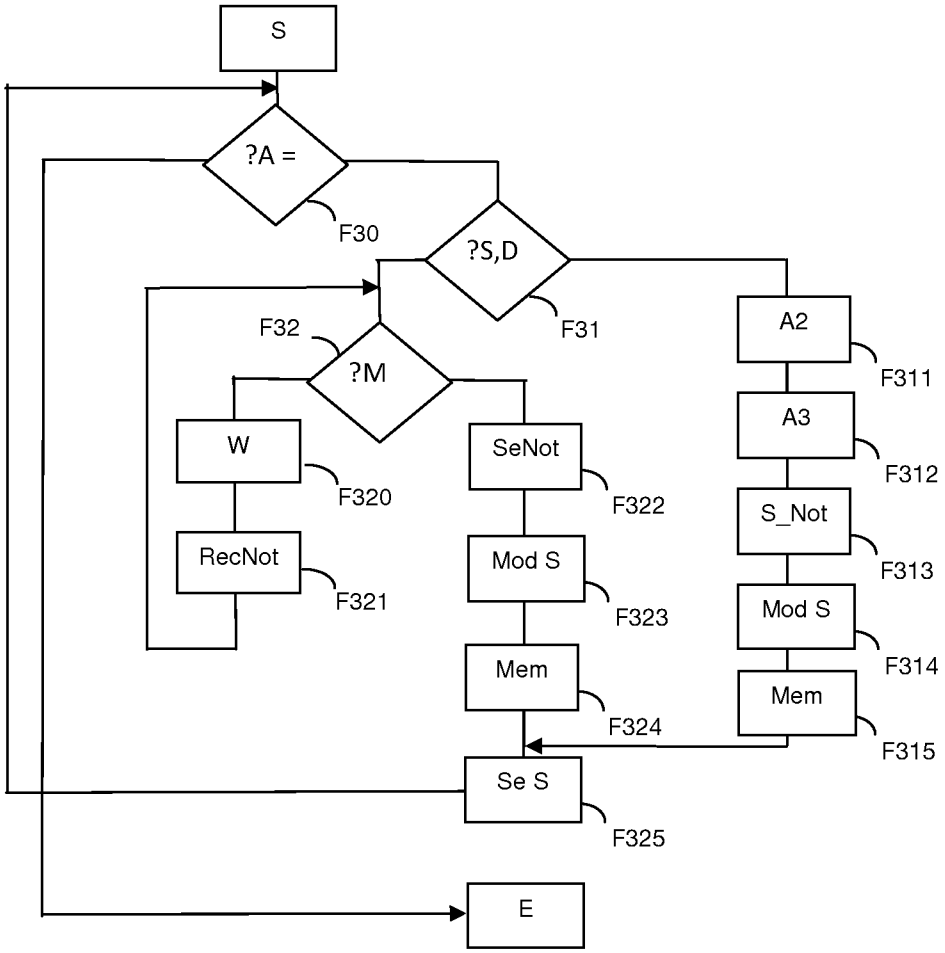

S
?A =
F30
?S,D
F31
F32
?M
W
F320
RecNot
F321
SeNot
F322
Mod S
F323
Mem
F324
Se S
F325
E
A2
F311
A3
F312
S_Not
F313
Mod S
F314
Mem
F315

Fig. 4b

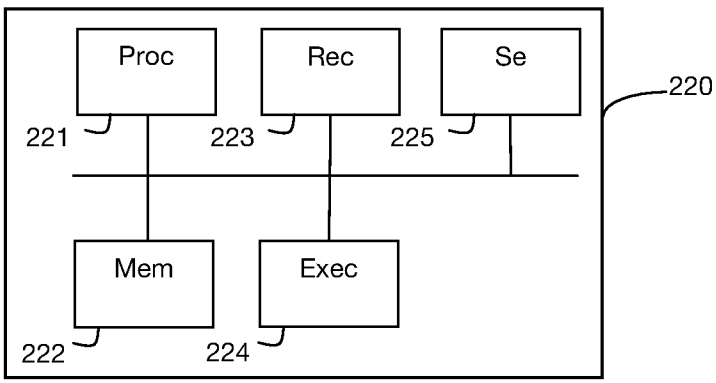

Proc
Rec
Se
220
221
223
225
Mem
Exec
222
224

Fig. 6

# INTERNATIONAL SEARCH REPORT

International application No
PCT/FR2016/051466

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/445 G06F9/455 G06F9/50 H04L29/08
ADD.

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F H04L

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X | Loic Letondeur: "Planication pour la gestion autonomique de l'élasticité d'applications dans le cloud", Thèse DE L'UNIVERSITÉ DE GRENOBLE, L'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique, 7 April 2015 (2015-04-07), pages 1-221, XP055261089, Retrieved from the Internet: URL:https://hal.archives-ouvertes.fr/tel-01140128/document [retrieved on 2016-03-24] pages 9-179 | 1-14 |

-----

-/--

[X] Further documents are listed in the continuation of Box C.

[ ] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

| Date of the actual completion of the international search | Date of mailing of the international search report |
|---|---|
| 1 August 2016 | 11/08/2016 |

| Name and mailing address of the ISA/ | Authorized officer |
|---|---|
| European Patent Office, P.B. 5818 Patentlaan 2<br>NL - 2280 HV Rijswijk<br>Tel. (+31-70) 340-2040,<br>Fax: (+31-70) 340-3016 | Breche, Philippe |

2

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X | ABID RIM ET AL: "Verification of a Dynamic Management Protocol for Cloud Applications", 15 October 2013 (2013-10-15), CORRECT SYSTEM DESIGN; [LECTURE NOTES IN COMPUTER SCIENCE; LECT.NOTES COMPUTER], SPRINGER INTERNATIONAL PUBLISHING, CHAM, PAGE(S) 178 - 192, XP047040181, ISSN: 0302-9743 ISBN: 978-3-642-28871-5 abstract; figures 1-4 paragraphs [0001] - [03.1], [03.4] - [0005] ----- | 1-14 |
| Y | XAVIER ETCHEVERS ET AL: "Automated Configuration of Legacy Applications in the Cloud", UTILITY AND CLOUD COMPUTING (UCC), 2011 FOURTH IEEE INTERNATIONAL CONFERENCE ON, IEEE, 5 December 2011 (2011-12-05), pages 170-177, XP032090829, DOI: 10.1109/UCC.2011.32 ISBN: 978-1-4577-2116-8 cited in the application abstract; figures 1-3 paragraphs [000I] - [00VI] ----- | 1-14 |
| Y | MARC LÃ CR GER ET AL: "Reliable Dynamic Reconfigurations in a Reflective Component Model", 23 June 2010 (2010-06-23), COMPONENT-BASED SOFTWARE ENGINEERING, SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, PAGE(S) 74 - 92, XP019142813, ISBN: 978-3-642-13237-7 abstract; figures 2-3 ----- | 1-14 |

2

# RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/FR2016/051466

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
INV. G06F9/445 G06F9/455 G06F9/50 H04L29/08
ADD.

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

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
G06F H04L

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

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

C. DOCUMENTS CONSIDERES COMME PERTINENTS

| Catégorie* | Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents | no. des revendications visées |
|---|---|---|
| X | Loic Letondeur: "Planication pour la gestion autonomique de l'élasticité d'applications dans le cloud", Thèse DE L'UNIVERSITÉ DE GRENOBLE, L'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique, 7 avril 2015 (2015-04-07), pages 1-221, XP055261089, Extrait de l'Internet: URL:https://hal.archives-ouvertes.fr/tel-01140128/document [extrait le 2016-03-24] pages 9-179 ----- -/-- | 1-14 |

[X] Voir la suite du cadre C pour la fin de la liste des documents

[ ] Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

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

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

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

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

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

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

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

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

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

| Date à laquelle la recherche internationale a été effectivement achevée | Date d'expédition du présent rapport de recherche internationale |
|---|---|
| 1 août 2016 | 11/08/2016 |

| Nom et adresse postale de l'administration chargée de la recherche internationale | Fonctionnaire autorisé |
|---|---|
| Office Européen des Brevets, P.B. 5818 Patentlaan 2<br>NL - 2280 HV Rijswijk<br>Tel. (+31-70) 340-2040,<br>Fax: (+31-70) 340-3016 | Breche, Philippe |

2

Formulaire PCT/ISA/210 (deuxième feuille) (avril 2005)

C(suite). DOCUMENTS CONSIDERES COMME PERTINENTS

| Catégorie* | Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents | no. des revendications visées |
|---|---|---|
| X | ABID RIM ET AL: "Verification of a Dynamic Management Protocol for Cloud Applications", 15 octobre 2013 (2013-10-15), CORRECT SYSTEM DESIGN; [LECTURE NOTES IN COMPUTER SCIENCE; LECT.NOTES COMPUTER], SPRINGER INTERNATIONAL PUBLISHING, CHAM, PAGE(S) 178 - 192, XP047040181, ISSN: 0302-9743 ISBN: 978-3-642-28871-5 abrégé; figures 1-4 alinéas [0001] - [03.1], [03.4] - [0005] | 1-14 |
| Y | XAVIER ETCHEVERS ET AL: "Automated Configuration of Legacy Applications in the Cloud", UTILITY AND CLOUD COMPUTING (UCC), 2011 FOURTH IEEE INTERNATIONAL CONFERENCE ON, IEEE, 5 décembre 2011 (2011-12-05), pages 170-177, XP032090829, DOI: 10.1109/UCC.2011.32 ISBN: 978-1-4577-2116-8 cité dans la demande abrégé; figures 1-3 alinéas [000I] - [00VI] | 1-14 |
| Y | MARC LÃ CR GER ET AL: "Reliable Dynamic Reconfigurations in a Reflective Component Model", 23 juin 2010 (2010-06-23), COMPONENT-BASED SOFTWARE ENGINEERING, SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, PAGE(S) 74 - 92, XP019142813, ISBN: 978-3-642-13237-7 abrégé; figures 2-3 | 1-14 |