L'apport de L'IDM dans l'administration du cloud computing
- Cas de l'autoscaling -

Faiza Chekaoui
Division de recherche et Développement sur les Réseaux
Centre de recherche sur l'Information Scientifique et Technique
Ben Aknoun, Algiers
Email : chekaoui@dtri.cerist.dz

***Résumé*—L'informatique utilitaire a toujours été l'ambition de beaucoup de chercheurs, l'arrivée du cloud computing peut être considérée comme une avancée importante pour sa réalisation. En effet obtenir de la puissance informatique comme on obtiendrait de la puissance électrique en branchant simplement un appareil dans une prise de courant semblait jusque là quelque chose d'irréalisable. La mise en place des clouds et surtout leurs vulgarisations (comparés aux grilles de calcul qu'on associe souvent aux scientifiques avertis) y a largement contribué coté des technologies de virtualisation qui permettent de donner l'illusion de capacités informatiques infinies. Il reste maintenant, à rendre cette nouvelle manière de consommer les ressources de calcul et de stockage la plus transparente possible. Le concept d'autonomie des systèmes informatiques peut être exploité pour ce besoins précisément. De manière concise, il vise à les doter des moyens nécessaires pour devenir les plus indépendants possible. Dans le présent article nous nous appuyons sur une approche de modélisation pour traiter d'un cas particulier de l'autonomie des systèmes informatiques qui est la mise à l'échelle automatique appelée autoscaling ou selfscaling. Nous exposons notre approche pour appuyer l'administration autonome du cloud computing libérant ainsi ses gestionnaires et ses utilisateurs de tâches routinières qui peuvent se compliquer avec l'offre diversifiée du marché actuel.**

***Keywords*—cloud computing, autoscaling, selfscaling, modelling, MDE, MDA.**

I. Introduction

Après quelques hésitations, l'adoption du cloud computing se généralise et touche plus d'entreprises. Bon nombre de décideurs ont fini par franchir le pas, motivés d'abord par l'augmentation perpétuelle des demandes sur les infrastructures et les applications auxquelles il devient de plus en plus difficile de faire face. Et ensuite par variation de ces demandes qui rend leur prédiction et leur satisfaction compliquées sans de réels risques financiers. C'est ainsi, que malgré les craintes qui subsistent autour de la sécurité et de la gestion inhérente à cette nouvelle offre, son admission ne fait aucun doute ; Gartner [1] déclare à ce propos que l'avènement du cloud a causé une augmentation de 25% des dépenses annuelles pour l'informatique en 2012 comparé à 2008 avec pas moins de 60% des serveurs achetés destinés à la virtualisation donc à la cloudification.

Plusieurs acteurs, de différentes tailles dont Amazon, Google et Microsoft pour n'en citer que cela, ont participé à cette offre. Ces acteurs évoluant généralement indépendamment les uns des autres, nous proposent aujourd'hui des offres diversifiées et hétérogènes. Des efforts pour promouvoir l'interopérabilité et la capitalisation des savoir-faire dans ce domaine ont été consentis, le présent travail s'inscrit dans cette optique en proposant une approche de modélisation pour certains aspects de l'interaction des administrateurs avec ces nouvelles plateformes.

Actuellement, de nombreux travaux se sont penchés sur cette problématique, quelques uns d'entre eux ayant opté pour des approches de modélisation. Cependant la plupart ont abordé les phases de déploiement et d'approvisionnement, il reste encore beaucoup de travail a faire concernant d'autres aspects de l'administration (aspects sécurité, aspects contractuels, etc). L'autoscaling est l'un de ces aspects, il fut intégré récemment dans certaines plateformes cloud, c'est cet aspect auquel nous nous sommes intéressés dans notre travail. L'objet étant, d'introduire une des approches de modélisation, qui est l'IDM (pour Ingénierie Dirigée par les Modèles) dans la prise en charge de cette facette de l'administration cloud.

Ce travail est présenté dans le reste de cet article organisé comme suit, l'approche IDM est d'abord présentée avant de détailler l'autoscaling et ses caractéristiques dans le cloud computing. La troisième partie est quand à elle, consacrée à notre approche pour introduire l'IDM dans la conception et l'implémentation de l'autoscaling. Une discussion succède à ces parties où les perspectives donnent la vue future sur la continuation du présent travail avant de conclure l'article.

II. L'approche IDM

L'Ingénierie Dirigée par les Modèles (IDM), en anglais Model-Driven Engineering, est une approche d'ingénierie générative par laquelle tout ou partie d'une application informatique est générée à partir de modèles [1]. La nouveauté dans l'approche est qu'elle nous fait passer de modèles dits contemplatifs - servant essentiellement à des fins de compréhension et de communication - vers des modèles productifs qui sont au centre de la production effective du logiciel [2].

L'approche IDM repose sur la séparation et le tissage (recomposition) des aspects, ces aspects peuventreprésenter des préoccupations qui apparaissent lors du développement, de la maintenance et de l'évolution des systèmes. MDA (Model Driven Architecture en anglais), qui est l'implémentation de

1. https ://www.gartner.com/doc/914826

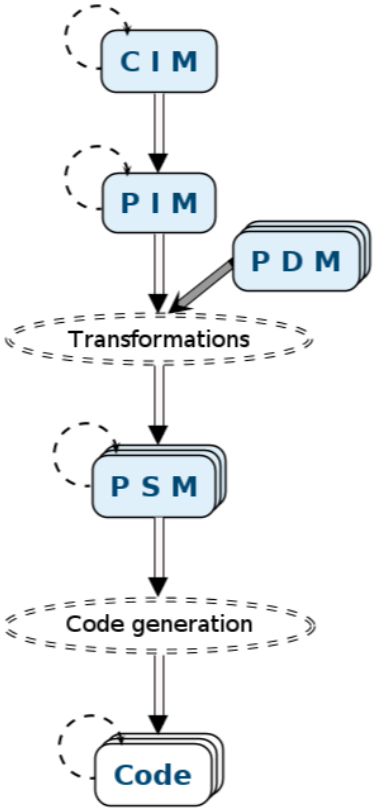

FIGURE 1. Processus MDA

l'approche IDM par l'OMG [2], est considérée sans conteste l'initiative la plus marquante de l'IDM. C'est aussi celle qui nous intéresse puisque elle sépare les aspects dépendants, des aspects indépendants des plateformes.

En effet, l'objectif de notre étude étant de faire face à l'hétérogénéité de l'offre cloud computing tout en capitalisant le savoir-faire des administrateurs, MDA semble être un candidat idéal pour aboutir à nos fins.

Brièvement, nous pouvons dire que MDA se base sur trois principaux concepts, le modèle (niveau M1), le méta-modèle (Niveau M2) et le métaméta-modèle (Niveau M3) liés entre eux par des relations de conformité qui permettent de valider les modèles utilisés pour représenter le système réel (niveau M0). L'idée de base de MDA étant de séparer les spécifications fonctionnelles d'un système des détails de son implémentation sur une plateforme donnée, ces concepts sont utilisés pour décrire plusieurs types de modèles qui mènent à la production du logiciel. Ce sont les modèles indépendants de l'informatisation appelés CIM (Computational Independent Model), les modèles indépendants des plateformes appelés PIM (Platform Independent Model), et les modèles spécifiques à une plateforme appelés PSM (Platform Specific Model). Ces derniers sont générés grâce à des transformations appliquées aux PIMs en se basant sur des PDM (Platform Description Model) [1] comme c'est illustré sur la figure 1.

2. Object Management Group - http ://www.omg.org

III. L'AUTOSCALING DANS LE CLOUD

L'autoscaling est une caractéristique des systèmes informatiques autonomes. Ce sont des systèmes qui s'inspirent du fonctionnement du système nerveux de l'être humain pour se doter de capacités d'auto-gestion. L'initiative lancé par IBM en 2001 définit déjà quatre propriétés générales qui constituent l'auto-gestion, ce sont l'auto-configuration, l'autoréparation, l'auto-optimisation et l'auto-protection. Depuis l'initiative d'IBM un intérêt grandissant pour ce domaine a fait rallonger la liste des propriétés que l'on note maintenant Self-* [3].

Concrètement, l'autoscaling consiste à redimensionner l'allocation des ressources de manière automatique afin de répondre aux besoins réels des applications. Il permet, dans des milieux cloud computing, de tirer avantage de l'élasticité et du modèle de payement pay-per-use qu'offre ce dernier. A titre d'exemple, les applications web qui font de plus en plus recourt aux solutions cloud, sont caractérisées par la forte variation des demandes d'accès. C'est notamment le cas des sites spécialisés dans le commerce en ligne et qui connaissent des pics importants pendant certaines périodes de l'année alors que l'accès est modéré pendant le reste du temps.

Permettre aux administrateurs de ces applications d'ajuster les ressources à ces variations tout en ayant la liberté de s'approvisionner chez différents four- nisseurs sans être contraint à réécrire à chaque fois les opérations d'approvisionnement, de déploiement et d'autoscaling est un gain considérable.

Pour rappel, la pile architecturale de l'offre Cloud Computing comprend trois principaux modèles de service qui correspondent chacun à un niveau d'abstraction : l'Infrastructure en tant que service(IaaS), la Plate-forme en tant que service (PaaS) et le Logiciel en tant que service (SaaS). L'implémentation d'un service d'autoscaling manipule des ressources dont le type dépende de la couche où l'on se positionne. Il s'agira de CPU de Mémoire et d'espace de stockage pour l'IaaS alors que pour le PaaS la ressource peut consister en des instances de serveurs (avec des configurations particulières). Au niveau SaaS, il sera plutôt question de l'aptitude des logiciels à se mettre à l'échelle pour s'adapter aux redimensionnements qui s'opèrent aux niveaux sous-jacents.

De manière générale, activer l'autoscaling auprès d'un fournisseur qui propose ce service, revient à définir un ensemble de ressources dont la cardinalité varient entre une valeur minimale et une valeur maximale. Certains indicateurs sont alors surveillés en permanence et des alertes sont déclenchées quant leurs valeurs changent ou atteignent des seuils prédéfinis. Les règles qui régissent ces événements sont attribuées aux groupes d'autoscaling pour décider de leurs évolutions à travers le temps. Une règle peut ressembler à ceci : *si le* ***temps de réponse moyen*** *est supérieur à* ***10s*** *alors* ***augmenter de 2*** *le nombre d'instances du serveur*

IV. ETAT DE L'ART

Selon [4], les solutions d'autoscaling peuvent être classées suivant le mécanisme d'élasticité qu'elles emploient. Ces mécanismes différent de par les champs qu'elles couvrent, les stratégies qu'elles adoptent, les objectifs qu'elles visent ou les méthodes qu'elles emploient, comme nous pouvons le voir sur la figure 2.

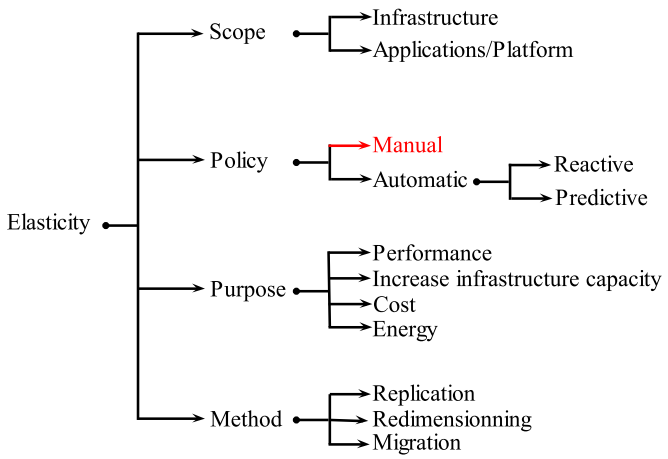

FIGURE 2. Classification proposed by Gallante [2012]

Il existe des travaux de recherche qui ont des points de similitude avec notre étude, en les examinant de près nous constatons que la grande majorité proposent des solutions qui opèrent au niveau IaaS du Cloud Computing implémentant des stratégies automatiques réactives et procèdent à la réplication des ressources aux moments des fortes demandes sur le service hébergé. La plateforme Claudia [5], le travail de Lim et al. [6], les projets [7], Tide [8], CHP [9] et Elastic Site [10] s'inscrivent tous dans ce groupe. Alors que nous retrouvons des travaux ayant employé des techniques pour prédire le comportement de la charge de travail sur le système, ce qui permet à priori d'anticiper les bonnes décisions concernant la mise à l'échelle des ressources. Ces travaux sont classés dans les stratégies automatiques/prédictives et comptent parmi eux [11], [12], [13] et [14].

Pour bénéficier pleinement de cette infrastructure élastique, les couches supérieures du cloud doivent être sensibles à cette propriété. Et en effet, certaines recherches ont présenté des mécanismes dont l'objectif principal est de permettre le développement d'applications flexibles, adaptables aux environnements cloud et qui tirent avantage de l'autoscaling. Nous retrouvons pour le niveau SaaS, des exemples comme, [15], [16] et [17] qui proposent des environnements de travail qui facilitent ce genre de développement. Toujours au niveau SaaS, le rendement des applications de streaming (diffusion audio vidéo) s'est vu sensiblement amélioré grâce à l'élasticité des clouds comme en donnent la preuve [18] et [19].

Le niveau PaaS a connu moins d'engouement que les autres niveaux, les travaux qui lui sont consacrés sont moins nombreux mais nous pouvons citer Aneka [20], un framework qui vise l'extension transparente des capacités locales disponibles dans un Desktop Grid [3] vers des ressources Cloud. Les décisions d'extension sont prises sur la base d'une estimation du temps d'exécution par l'utilisateur, si les ressources locales ne suffisent pas pour respecter cette estimation, le système recourt à ces ressources additionnelles. L'étude la plus proche de la notre est vraisemblablement la proposition de Martin et al. [21]. Elle consiste à traiter le problème de la gestion des services élastiques dans des environnements Cloud. Ils étendent pour cela, un framework à base d'agents pour la gestion des services en l'interfaçant avec les systèmes de gestion de ressources intégrés dans les plateformes cloud disponibles. La solution adhère à la logique de la boucle MAPE-K [4] de l'informatique autonome, en implémentant des agents qui jouent le rôle de *Capteur*, le rôle d'*Acteur* ou le rôle d'*Actionneur*. Les Capteurs collectent les événements ; les Acteurs décident des actions et les planifient ; les Actionneurs appliquent les modifications décidées en exécutant le plan défini par les Acteurs. Ces derniers s'appuient sur des informations comme la charge de travail, les performances ou les coûts engendrés afin de déterminer les VMs les plus appropriées pour gérer le surplus de charge. Quelques solutions propriétaires comme Amazon EC2 [22] et Microsoft Azure [23] ont intégré un service d'autoscaling, des solutions open-sources ont aussi emboité le pas, même si cela s'est fait que très tardivement (juin 2013 pour Eucalyptus et openNebula). Cependant, aucun consensus ne semble les réunir, le client reste de ce fait lié à son fournisseur tant que ce dernier reste fermé sur lui même en adoptant des solutions propriétaires. Hors l'autonomie qui tente de minimiser l'intervention des administrateurs dans la gestion des systèmes est confrontée à une impasse lorsqu'il est question d'opérer dans des environnements cloud hétérogènes. La solution mise en place pour un environnement particulier ne peut fonctionner aisément dans un autre environnement. Nous constatons de ce fait que l'hétérogénéité et la capitalisation des savoir-faire ne forment pas les objectifs des travaux que nous venons de citer.

D'un autre coté, il existe des projets qui proposent de lever la complexité lié à la gestion cloud par l'introduction des approches de modélisation. Ces premiers travaux ont tout naturellement adresser les problématiques de base, telles que le développement et le déploiement d'applications et parfois l'approvisionnement en ressources. Il est donc intéressant, avec la même approche, d'essayer de résoudre d'autres problématiques, comme l'autoscaling bien sûr mais aussi l'autonomie en général. Nous citerons à titre d'exemple, PIM4Cloud [24], CloudML [25] et MODAClouds [26] sont tous des projets qui ont opté pour l'utilisation des modèles mais concernent plus particulièrement les étapes d'approvisionnement et de déploiement tout en prévoyant d'intégrer les facilités de répartition de charge et d'autoscaling dans leurs perspectives.

Comme nous venons de le voir, le service d'autoscaling ayant été intégré que récemment dans les plateformes cloud. À notre connaissance, sa modélisation n'a pas encore été réalisée. Nous nous sommes donc fixé comme but de proposer une plateforme à base de modèles qui supporte l'autoscaling.

V. PLATEFORME D'AUTOSCALING À BASE DE MODÈLES

La plateforme que nous mettons en place doit être capable de réaliser une opération d'autoscaling à chaque fois que cela est nécessaire, c'est-à-dire qu'elle ajuste (augmente ou diminue) le nombre d'instances de l'application à la demande. L'utilisation de MDA nous permet en fait de répondre aux exigences formulées précédemment : le savoir-faire des administrateurs est capturé dans des modèles (PIM) indépendants des plateformes cibles. Aussi la génération à la demande des modèles dépendants des plateformes grâce aux transformations assure la portabilité entre ces plateformes.

3. Plateformes qui utilisent les cycles inactifs des PCs de bureaux pour atteindre de grandes puissance de calcul

4. Monitor, Analyse, Plan, Execute - Knowledge

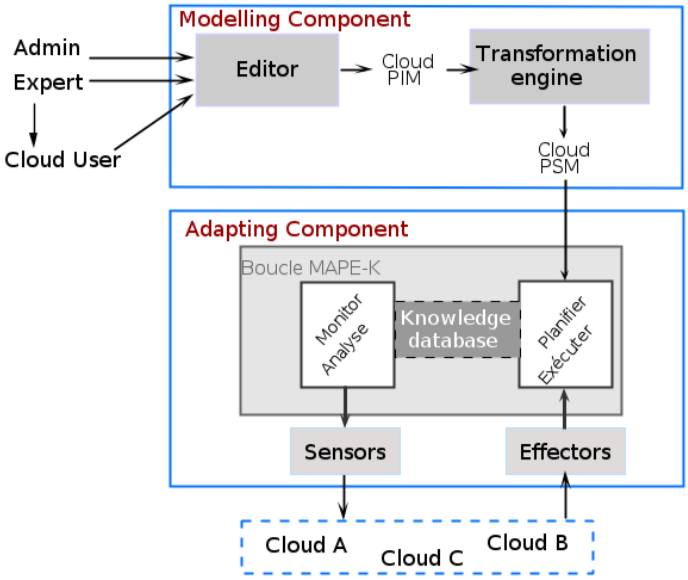

FIGURE 3. Platform architecture

De cette description se dégagent deux environnements distincts et complémentaires dans notre plateforme, l'environnement de modélisation et l'environnement d'adaptation, l'architecture globale qui englobe ces deux environnements est illustrée sur la figure 3.

A. L'environnement de modélisation assiste l'utilisateur dans la phase de modélisation où il est possible de distinguer les deux niveaux d'abstraction PIM et PSM que nous avons évoqués et un moteur de raffinement qui procède aux transformations pour passer du premier modèle au second. Des templates réalisés par des utilisateurs experts sont aussi préparés pour servir les utilisateurs ayant moins de connaissances dans le domaine.
B. L'environnement d'adaptation est la partie responsable d'assurer la fonction d'autoscaling en interprétant le modèle généré précédemment. C'est en fait une interface qui interagit directement avec des environnements cloud.

L'environnement respecte la boucle MAPE-K en réservant :

— Un composant qui procède à la surveillance et l'analyse des alarmes programmées et qui transmets les valeurs collectées par les capteurs ;
— Un composant chargé de stocker les connaissances qui servent à la prise des décisions de redimensionnement ;
— Un composant qui se charge de planifier et d'exécuter des actions d'autoscaling par l'emploi d'actionneurs.

L'implémentation de l'environnement d'adaptation est fortement lié aux APIs proposées par les fournisseurs cloud. Au mieux, il faut se baser sur les opérations de base telles que la création et la suppression des VMs, le déploiement d'images, etc... Mais certaines plateformes cloud sont très matures, elle proposent en plus, des fonctions pour faciliter la configuration des groupes d'autoscaling, l'environnement d'adaptation doit tirer avantage de ces riches APIs.

VI. AUTOSCALING MODELS

Comme nous l'avons mentionné précédemment, MDA préconise l'utilisation des Modèles CIM, PIM et PSM, nous décrivons nos modèles dans les sections qui suivent.

A. CIM metamodel

A partir de plusieurs définitions de l'autoscaling nous avons constituer un modèle CIM, il décrit cinq entités principales : le groupe, la règle, l'alarme, l'action et la ressource cloud.

Les groupes : Un groupe d'autoscaling est composé d'un nombre variable de ressources cloud alloués à un service donné. Il est caractérisé par une configuration qui sert à lancer les nouvelles instances du groupe ;
Les règles : Une règle d'autoscaling sont des contraintes qui expriment la stratégie d'approvisionnement du groupe, un ensemble de règles reflète les conditions dans lesquels, une action d'ajuster la taille du groupe est prise ;
Les alarmes : Une règle est basé sur les changements d'état d'une alarme. Les propriétés d'une ressource, comme le taux d'utilisation de la CPU ou le temps de réponse sont contrôlés, leurs variations sont immédiatement signalés et analysés (comparés à la valeur d'un seil donné) et si nécessaire, une alarme dédiée est enclenchée ;
Les actions : Une action permet de réajuster la taille d'un groupe, il peut s'agir de l'ajout ou du retrait de ressources ; dans les deux cas le nombre de ressources concernées doit être précisé ;
Les ressources : Les ressources cloud les plus répandues sont la CPU, l'espace de stockage, la RAM ou une instance de VM mais elle peut aussi représenter d'autres ressources telles qu'un bloc de stockage, une ressource réseau, etc... . Les informations collectées sur ces ressources permettent d'améliorer la prise de décision concernant l'autoscaling.

B. PIM metamodel

La précédente description est la principale partie du modèle CIM, elle nous a servi à élaborer notre métamodèle PIM illustré à la figure 4, il reprend les concepts de base tout en exprimant les liens entre eux. Brievement, lire ceci un groupe d'autoscaling

C. PSM metamodels

Les concepts qui composent notre métamodèle PIM sont manipulés dans la-plupart des plateformes cloud avec les mêmes termes ou des termes équivalents. Mais dans certains cas, un même terme peut avoir des significations complètement différentes. Nous avons donc procéder à l'identification des concepts communs et avons noté les différences dans certaines plateformes cloud. C'est pour ces raisons que chaque plateforme cible a son propre métamodèle PSM qui contient les détails techniques spécifiques à la plateforme considérée. La figure 5 par exemple, illustre les concepts d'autoscaling qu'on retrouve dans le cloud Eucalyptus [5], ce metamodèle nous sert à générer des modèles PSM valide pour Eucalyptus à partir d'un modèle PIM donné.

5. https://www.eucalyptus.com

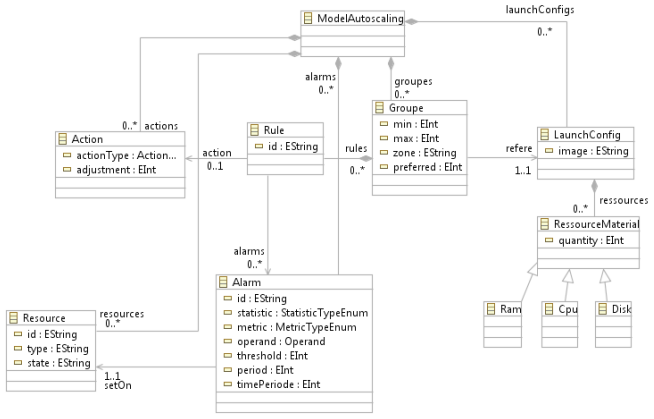

FIGURE 4. PIM

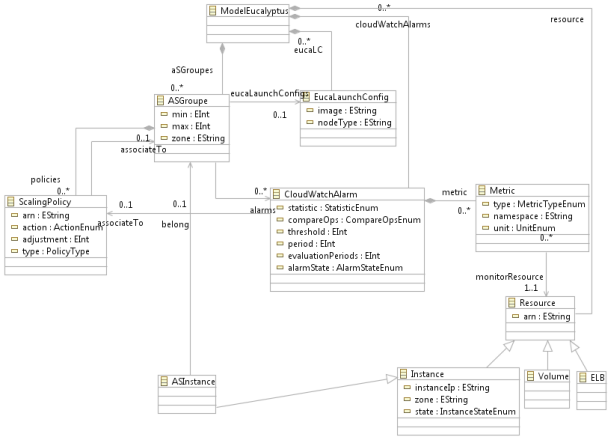

FIGURE 5. PSM - Eucalyptus

VII. MODEL TRANSFORMATION

Le passage d'un modèle conforme au métamodèle PIM qui expriment les besoins d'autoscaling vers un modèle conforme au métamodèle PSM, se fait à l'aide de transformations de type M2M[6]. Et afin d'agir sur la plateforme cloud ciblée via les APIs, nous employons une autre transformation du modèle PSM, cette seconde transformation est de type M2T[7]. Nous décrivons le processus de transformation avec ces deux phases dans la suite de cette section.

A. Transformation PIM vers PSM

La première transformation est assurée par un ensemble de règles qui expriment la correspondance entre les concepts issus de différents modèles. Plusieurs concepts source peuvent être combinés pour produire un seul concept cible, ou au contraire, un concept source peut être éclaté pour former plusieurs concepts cible comme nous l'illustrons dans les figures 6 and 7.

Les transformations sont réalisées en langage ATL dont le listing 1 représente un extrait.

6. Model to Model
7. Model to Test

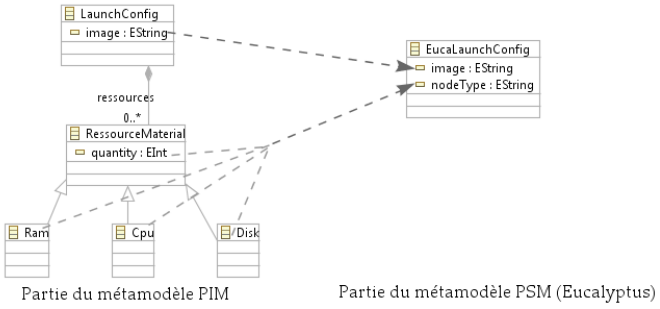

FIGURE 6. PIM to PSM - Combining concepts

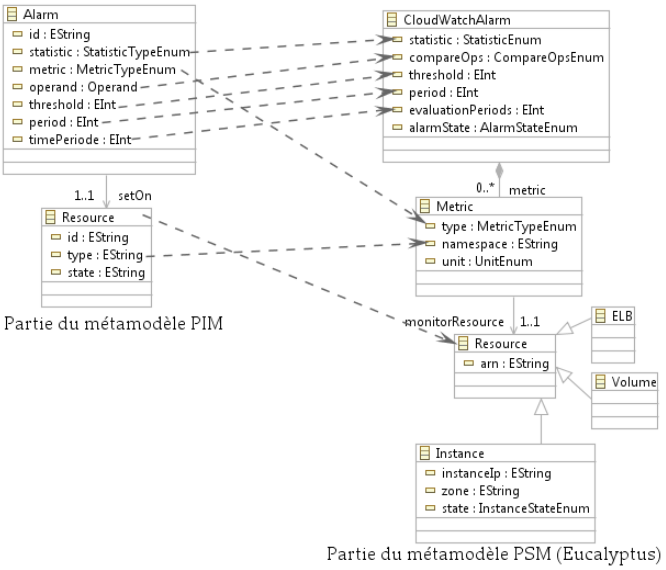

FIGURE 7. PIM to PSM - Explosing concept

Listing 1. Transformation MM-PIM vers MM-PSM(ATL)

```
-- @atlcompiler atl2006
-- @nsURI MM=http://autoscaling/1.1
-- @nsURI MM1=http://eucalyptus/1.1
module Autoscaling2Eucalyptus;
create OUT: MM1 from IN: MM;
-- HELPERS --
helper context MM!LaunchConfig def: getType(ram: Integer,
     cpu: Integer, disk: Integer):
 String = if cpu.first() = 1 then
                  if ram.first() = 128 then 'm1small'
                  else 'c1medium'
                  endif
           else if cpu.first() = 2 then
                        if ram.first() = 512
                        then 'm1large'
                        else 'm1xlarge'
                        endif
                  else 'c3xlarge'
                  endif
           endif;
.
.
.

rule LaunchConfig2EucaLaunchConfig {
 from lg: MM!LaunchConfig
 to  eucalg: MM1!EucaLaunchConfig mapsTo lg (
           name <- lg.name,
           image <- lg.image,
           nodeType <- lg.getType(lg.ressources -> select(e
                 | e.oclIsKindOf(MM!Ram)) ->
```

```
                        collect(e | e.quantity), lg.
                             ressources -> select(e |
                        e.oclIsKindOf(MM!Cpu)) -> collect(e
                             | e.quantity), lg.ressources ->
                        select(e | e.oclIsKindOf(MM!Disk))
                             -> collect(e | e.quantity)))
}
.
.
.
      -------------------End Rule --------------------------
```

B. Transformation PSM vers code API

Le modèle PSM issu de la première étape subit une seconde transformation de type M2T. Le listing 2 est un extrait du programme qui parcourt le modèle afin d'en extraire l'information utile. Un template est alors enrichis avec ces données afin de pouvoir adresser le cloud cible via son API. Le résultat de cette étape nous permet de mettre en place un groupe d'autoscaling conforme au besoins exprimés dans le modèle PIM de départ.

Listing 2. Transformation modèle PSM vers Code API.

```
[comment encoding = UTF-8 /]
[module generate('http://eucalyptus/1.1')]
[template public generateElement(aModelEucalyptus :
     ModelEucalyptus)]
[comment @main/]

[file ('myFile.sh', false, 'UTF-8')]
.
.
.
[for (policy: ScalingPolicy | aModelEucalyptus.aSGroupes.
     policies)]
eucascale-put-scaling-policy [policy.name/] --auto-scaling-
     group [aModelEucalyptus.aSGroupes.name/] --adjustment=[
     policy.adjustment/] --type [policy.type/]
       [for (alarm : CloudWatchAlarm | aModelEucalyptus.
            cloudWatchAlarms)]
         [if (alarm.associateTo = policy)]
euwatch-put-metric-alarm [alarm.name/] --metric-name [alarm.
     metric.type/] --unit [alarm.metric.unit/] --namespace
     "[alarm.metric.namespace/]" --statistic [alarm.
     statistic/] --period [alarm.period/] --threshold [alarm
     .threshold/] --comparison-operator [alarm.compareOps/]
     --dimensions "AutoScalingGroupName=[aModelEucalyptus.
     aSGroupes.name/]" --evaluation-periods [alarm.
     evaluationPeriods/] --alarm-actions [policy.arn/]
           [/if]
       [/for]
[/for]
.
.
.
[/file]
[/template]
```

VIII. Conclusion

L'administration des ressources cloud se complexifie avec la taille grandissante des plateformes cloud et plus encore quand celles-ci sont hétérogènes. Le fournisseur, tout autant que les clients gagnent considérablement en rendant les procédures répétitives d'administration moins dépendantes de leur intervention directe. Dans le travail que nous venons de présenter, nous nous sommes penchés sur le service d'autoscaling dans le cloud computing pour évaluer l'apport des modèles dans l'administration autonome de ces environnements. Nous découvrons également que beaucoup d'aspects se prêtent bien à la modélisation, des formats des applications aux templates SLA, en passant par l'authentification, les entrepôts de données rattachés aux machines virtuelles, les mécanismes de monitoring des applications.... L'expression de tous ces aspects et bien d'autres dans des formats compréhensibles par tous les acteurs aboutit assurément à une meilleure interopérabilité et à une portabilité des codes dans les environnements Cloud.

Au final, nous constatons qu'il est évident que les modèles peuvent apporter une amélioration au niveau de l'administration du Cloud et renforcer son autonomie vis à vis de l'intervention humaine, il faut seulement déterminer à quel niveau faut-il intervenir et comment harmoniser et intégrer les différents efforts de modélisation.

Références

[1] S. Diaw, R. Lbath, and B. Coulette, "Etat de l'art sur le développement logiciel basé sur les transformations de modèles," *Technique et Science Informatiques, Ingénierie Dirigée par les Modèles*, vol. 29, no. 4-5/2010, pp. 505–536, 2010.

[2] J. Estublier, J. Favre, J. Bézivin, and L. Duchien, "Action Spécifique CNRS sur l'Ingénierie Dirigée par les Modèles," CNRS, Tech. Rep. Mdd, 2005. [Online]. Available : http://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:Action+Sp\'{e}cifique+CNRS+sur+l'Ing\'{e}nierie+Dirig\'{e}e+par+les+Mod\'{e}les\#0

[3] R. Sterritt, M. Parashar, H. Tianfield, and R. Unland, "A concise introduction to autonomic computing," *Advanced Engineering Informatics*, vol. 19, no. 3, pp. 181–187, Jul. 2005. [Online]. Available : http://linkinghub.elsevier.com/retrieve/pii/S147403460500042X

[4] G. Galante and L. C. E. De Bona, "A Survey on Cloud Computing Elasticity," *2012 IEEE Fifth International Conference on Utility and Cloud Computing*, vol. 0, no. 1, pp. 263–270, Nov. 2012. [Online]. Available : http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6424959

[5] Claudia, "Projet Claudia," http ://www.reservoir-fp7.eu/index.php ?page=claudia. [Online]. Available : http://www.reservoir-fp7.eu/index.php?page=claudia

[6] H. Lim, S. Babu, J. Chase, and S. Parekh, "Automated control in cloud computing : challenges and opportunities," in *1st Workshop on Automated Control for Datacenters and Clouds*. ACM, 2009, pp. 13–18. [Online]. Available : http://dl.acm.org/citation.cfm?id=1555275

[7] A. Chazalet and F. D. Tran, "Self-scaling the Cloud to meet Service Level Agreements," *CLOUD COMPUTING* ..., no. c, pp. 116–121, 2010. [Online]. Available : http://www.thinkmind.org/index.php?view=article\&articleid=cloud_computing_2010_5_20_50060

[8] S. Meng, L. Liu, and V. Soundararajan, "Tide : Achieving Self-Scaling in Virtualized Datacenter Management Middleware," in *Proceedings of the 11th International Middleware Conference Industrial track on - Middleware Industrial Track '10*. New York, New York, USA : ACM Press, 2010, pp. 17–22. [Online]. Available : http://dl.acm.org/citation.cfm?id=1891722http://portal.acm.org/citation.cfm?doid=1891719.1891722

[9] J. Fitó, I. Goiri, and J. Guitart, "SLA-driven Elastic Cloud Hosting Provider," in *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*. Pisa : Ieee, Feb. 2010, pp. 111 – 118. [Online]. Available : http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5452504http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5452504

[10] P. Marshall, K. Keahey, and T. Freeman, "Elastic Site : Using Clouds to Elastically Extend Site Resources," in *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE, 2010, pp. 43–52. [Online]. Available : http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5493493

[11] J. Espadas, A. Molina, G. Jiménez, M. Molina, R. Ramírez, and D. Concha, "A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures," *Future Generation* ..., vol. 29, no. 1, pp. 273–286, Jan. 2013. [Online]. Available : http://linkinghub.elsevier.com/retrieve/pii/S0167739X1100210Xhttp://www.sciencedirect.com/science/article/pii/S0167739X1100210X

[12] H. Ghanbari, B. Simmons, M. Litoiu, and G. Iszlai, "Feedback-based optimization of a private cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 104–111, Jan. 2012. [Online]. Available : http://linkinghub.elsevier.com/retrieve/pii/S0167739X11001014

[13] N. Vasić, D. Novaković, and S. Miučin, "Dejavu : accelerating resource allocation in virtualized environments," *ACM SIGARCH Computer Architecture News - ASPLOS '12*, vol. 12, no. 1, pp. 423–436, 2012. [Online]. Available : http://dl.acm.org/citation.cfm?id=2151021http://infoscience.epfl.ch/record/169841/files/DejaVu-final.pdf?version=1

[14] B. Dougherty, J. White, and D. Schmidt, "Model-driven auto-scaling of green cloud computing infrastructure," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 371–378, Feb. 2012. [Online]. Available : http://linkinghub.elsevier.com/retrieve/pii/S0167739X11000902http://www.sciencedirect.com/science/article/pii/S0167739X11000902

[15] I. Neamtiu, "Elastic executions from inelastic programs," in *Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems (SEAMS '11)*, 2011, pp. 178–183. [Online]. Available : http://dl.acm.org/citation.cfm?id=1988008.1988033

[16] D. Rajan, A. Canino, J. A. Izaguirre, and D. Thain, "Converting a High Performance Application to an Elastic Cloud Application," *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, pp. 383–390, 2011.

[17] A. Raveendran, T. Bicer, and G. Agrawal, "A Framework for Elastic Execution of Existing MPI Programs," *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pp. 940–947, May 2011. [Online]. Available : http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6008941

[18] S. Vijayakumar, Q. Zhu, and G. Agrawal, "Dynamic Resource Provisioning for Data Streaming Applications in a Cloud Environment," *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pp. 441–448, Nov. 2010. [Online]. Available : http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5708483

[19] T. Knauth and C. Fetzer, "Scaling Non-elastic Applications Using Virtual Machines," *2011 IEEE 4th International Conference on Cloud Computing*, pp. 468–475, Jul. 2011. [Online]. Available : http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6008744

[20] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. a. F. De Rose, and R. Buyya, "CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software : Practice and Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011. [Online]. Available : http://doi.wiley.com/10.1002/spe.995

[21] P. Martin, A. Brown, W. Powley, and J. L. Vazquez-Poletti, "Autonomic management of elastic services in the cloud," *2011 IEEE Symposium on Computers and Communications (ISCC)*, pp. 135–140, Jun. 2011. [Online]. Available : http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5984006

[22] Amazon WS, "Amazon WS," http ://aws.amazon.com/. [Online]. Available : http://aws.amazon.com/

[23] F. Lardinois, "Microsoft Adds Auto Scaling To Windows Azure," 2013. [Online]. Available : http://techcrunch.com/2013/06/27/microsoft-adds-auto-scaling-to-windows-azure/

[24] E. Brandtzæ g, "CloudML A DSL for model-based realization of applications in the cloud," Master thesis, UNIVERSITY OF OSLO, 2012.

[25] E. Brandtzæ g, P. Mohagheghi, and S. Mosser, "Towards a domain-specific language to deploy applications in the clouds," in *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, 2012, pp. 213–218. [Online]. Available : http://www.thinkmind.org/index.php?view=article\&articleid=cloud_computing_2012_9_40_20200

[26] D. Ardagna and E. D. Nitto, "Modaclouds : A model-driven approach for the design and execution of applications on multiple clouds," *Modeling in Software …*, pp. 50–56, 2012. [Online]. Available : http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6226014