HAL
archives-ouvertes.fr

Vers un Outil de Configuration et de Déploiement pour les Nuages

Clément Quinton, Laurence Duchien

▶ **To cite this version:**

Clément Quinton, Laurence Duchien. Vers un Outil de Configuration et de Déploiement pour les Nuages. JLdP - Journée Lignes de Produits, Nov 2012, Lille, France. pp.83-94, 2012. <hal-00747319>

HAL Id: hal-00747319
https://hal.inria.fr/hal-00747319

Submitted on 31 Oct 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Vers un Outil de Configuration et de Déploiement pour les Nuages

Clément Quinton — Laurence Duchien

Inria Lille - Nord Europe
LIFL UMR CNRS 8022
University Lille 1, France
prenom.nom@inria.fr

RÉSUMÉ. *L'informatique dans les nuages est une tendance actuelle majeure pour répartir les traitements et les données de façon virtuelle sur des environnements d'exécution paramétrables. Le développement et le déploiement de logiciels pour les nuages proposent un nouveau challenge scientifique en termes d'expression et de prise en compte de la variabilité. En effet, l'informatique dans les nuages repose sur des principes d'hétérogénéité et d'élasticité, ce qui permet de nombreux choix de configuration et de dimensionnement. Les Modèles de Caractéristiques (*MC*) issus de l'approche Ligne de Produits Logiciels (LPL) sont une réponse possible pour gérer cette variabilité, préparer et dimensionner des configurations à déployer dans les nuages. Dans cet article, nous introduisons SALOON, un cadre logiciel d'expression de la variabilité et d'aide à la décision pour configurer et dimensionner des applications à déployer dans les nuages. Basé sur des ontologies et des* MCs *étendus, il prend en compte les aspects techniques et non-fonctionnels de l'application pour trouver un fournisseur de nuages qui correspond au mieux à la configuration de l'application.*

ABSTRACT. *Cloud Computing is a major trend in distributed computing environments enabling software virtualization on configurable runtime platforms. Development and deployment of Cloud software systems result in new challenges to express and manage their intrinsic variability. Many configuration and customization choices arise due to the heterogeneous and scalable aspect of the Cloud Computing paradigm. Features Model originating from Software Product Line (SPL) approach is one way to handle this variability, manage, create and deploy configuration in the Cloud. In this paper, we introduce* SALOON*, a framework to configure and describe variability for applications to be deployed in the Cloud. Based on ontologies and extended features models,* SALOON *takes application's technical and non-functional requirements into consideration to provide the most appropriate cloud solutions.*

MOTS-CLÉS : *Informatique dans les Nuages, Ligne de Produits Logiciels, Ontologie, Modèle de Caractéristiques Étendu*

KEYWORDS: *Cloud Computing, Software Product Line, Ontology, Extended Feature Model*

1. Introduction

L'informatique dans les nuages (*Cloud Computing*) est une tendance majeure dans les environnements d'informatique répartie qui permet d'accéder à des ressources virtuelles configurables (*e.g.*, serveurs, espace de stockage, réseaux) sous forme de services délivrés à la demande par les fournisseurs de Cloud [MEL 09]. Pour être accessible en tant que service (*Software-as-a-Service*, SaaS) sur le Cloud, une application peut être déployée soit sur l'infrastructure (*Infrastructure-as-a-Service*, IaaS) du fournisseur de Cloud, soit sur la plateforme qui va l'exécuter (*Platform-as-a-Service*, PaaS). Dans les deux cas, il faut configurer et dimensionner l'environnement d'exécution de l'application pour prendre en compte l'hétérogénéité et l'élasticité inhérentes au paradigme du Cloud Computing. En effet, il existe un grand nombre de ressources ayant des niveaux de fonctionnalité et des possibilités de dimensionnement hétérogènes parmi les différentes solutions de cloud disponibles (Amazon EC2, Jelastic, CloudBees, taille de l'instance, etc.). Lors d'un déploiement d'application sur un PaaS, il faut sélectionner un serveur d'application, la fréquence CPU nécessaire à l'exécution, la base de données à utiliser, sa capacité, etc. Il en va de même pour le déploiement sur un IaaS qui nécessite la configuration de toute la pile logicielle (système d'exploitation, librairies, serveurs d'applications) qui supporte l'application et qui s'exécute sur l'infrastructure du fournisseur de cloud. Cette variabilité en termes de configuration et de dimensionnement offre une multitude de possibilités de configurations qui sont généralement effectuées de manière ad hoc et sont sources d'erreurs lorsqu'elles sont réalisées à la main.

Nous pensons que cette sélection peut être en partie guidée et automatisée en utilisant des outils issus de l'approche Ligne de Produits Logiciels (LPL) [CLE 01, POH 05] et de définition d'ontologies [GRU 93]. Les LPLs permettent la description, la gestion et le développement de logiciels qui partagent un ensemble de caractéristiques communes mais qui diffèrent en certains points de variabilité, formant ainsi une famille de produits logiciels. Une approche bien connue pour modéliser la variabilité repose sur l'utilisation des Modèles de Caractéristiques (MC) proposés par [KAN 90] dans son étude *Feature Oriented Domain Analysis* (FODA). La variabilité des environnements de Cloud Computing peut ainsi être représentée par des MCs et le dimensionnement de la configuration est effectué par le biais d'attributs qui étendent le MC [BEN 05]. De plus, nous proposons d'utiliser les ontologies [GRU 93] comme modèle de données représentatif du domaine du Cloud Computing pour en décrire les concepts, les relations entre ces concepts et leur sémantique. Différentes caractéristiques de Cloud peuvent ainsi être capturées sous le même concept. En définissant le domaine, les points communs, les points de variabilité et la famille de produits à dériver, nous décrivons les principaux objectifs du processus d'ingénierie du domaine des LPLs [POH 05] appliqué au Cloud, et donnons une ébauche du processus d'ingénierie de l'application en expliquant le processus de configuration. De plus, nous proposons d'utiliser les ontologies comme modèle de données représentatif du domaine du Cloud Computing pour en décrire les concepts, les relations entre ces concepts et leur sémantique. Dans cet article, nous présentons donc une approche basée sur l'utilisation d'ontologies et de MCs étendus pour identifier le(s) fournisseur(s) de Cloud qui

permettent de déployer l'application et de choisir parmi les solutions possibles celle qui répond aux exigences techniques de la configuration et au dimensionnement de ces exigences.

La structure de l'article est la suivante. Dans la SEC. 2, nous décrivons quels sont les problèmes liés au déploiement dans le Cloud et les challenges auxquels font face les acteurs du Cloud. La SEC. 3 présente SALOON, l'approche basée sur les ontologies et les MCS pour choisir son fournisseur de Cloud. Dans la SEC. 4, nous donnons les détails d'implémentation et discutons des limitations de l'approche. La SEC. 5 évoque les travaux connexes tandis que la SEC. 6 conclut cet article.

2. Motivations & Challenges

Le déploiement d'une application dans le Cloud offre à son propriétaire de nombreux avantages : réduction des coûts, passage à l'échelle, haute disponibilité, etc. Cependant, la migration d'une application ou le développement d'un nouveau service dans le Cloud n'est pas trivial. En effet, il existe un grand nombre de possibilités au moment de choisir le fournisseur de Cloud capable de fournir l'environnement pour héberger l'application en tant que service. Ainsi, certains utilisateurs préfèrent déployer leur application vers une solution de Cloud qu'ils ont déjà utilisée mais qui ne correspond pas complètement aux exigences de l'application à déployer (*e.g.*, changer de type de base de données) plutôt que de se risquer à configurer un PaaS ou un IaaS pourtant plus adapté à leurs exigences. Le Cloud offre de nombreux choix de configuration et de dimensionnement tant au niveau de l'application à déployer que des environnements d'exécution IaaS ou PaaS paramétrables (FIG. 1). Il est donc très difficile de prendre les bonnes décisions face aux questions suivantes : faut-il privilégier un déploiement niveau PaaS ? IaaS ? Quel fournisseur choisir ? Comment être sûr qu'il représente une bonne solution pour les types d'applications et de déploiements souhaités ?

Figure 1. *Les différentes couches du Cloud Computing*[1]

À notre connaissance, il n'existe pas à l'heure actuelle d'outil d'aide à la configuration et au dimensionnement permettant de choisir finement le fournisseur de Cloud idéal, ou du moins le plus en adéquation avec l'environnement technique et non-fonctionnel de l'application. Aujourd'hui, ce choix repose sur les connaissances des

1. `http://blog.gravitant.com/wp-content/uploads/2012/07/CloudTechSpectrum_Vendors_v21.png`

experts en Cloud Computing et soulève le problème de la fiabilité et de l'exhaustivité de cette connaissance. Le cadre logiciel présenté dans cet article est *une* solution proposée pour concevoir un outil de configuration et de déploiement dans le Cloud. Il a pour objectif de relever deux challenges auxquels sont confrontés tous les acteurs concernés par un déploiement dans le Cloud :

$\mathcal{C}_1$: *Gestion de la variabilité technique et applicative.* Lors du déploiement de l'application dans le Cloud, il faut s'assurer de la compatibilité entre les exigences techniques de l'application et l'environnement technique des fournisseurs de Cloud *avant* le déploiement pour identifier un fournisseur adéquat.

$\mathcal{C}_2$: *Gestion du dimensionnement.* L'utilisateur doit pouvoir dimensionner les Qualités de Service en adéquation avec les exigences techniques et les ressources du Cloud pour pouvoir choisir plus précisément parmi les fournisseurs de Cloud potentiellement identifiés en $\mathcal{C}_1$.

3. SALOON, un Cadre Logiciel Basé sur les Modèles pour la Configuration et le Déploiement dans le Cloud

Pour faire face à ces deux challenges, nous proposons un outil de configuration et de déploiement appelé *SALOON*. Basé sur des ontologies et des MCs, SALOON permet *(i)* de spécifier une configuration technique pour l'application à déployer et *(ii)* de dimensionner cette configuration.

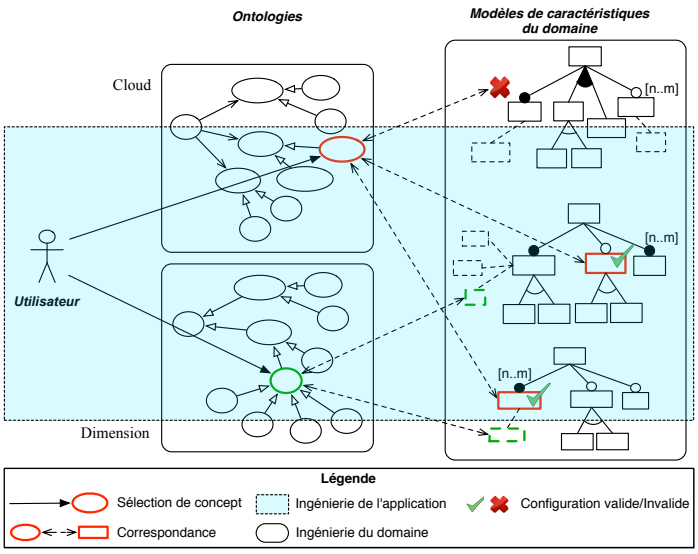


Figure 2. *Architecture de SALOON*

***Vue générale de* SALOON.** Les ontologies et les MCs de SALOON (FIG. 2) forment l'architecture de base et représentent l'ingénierie du domaine, tandis que la sélection de concepts et de caractéristiques constitue l'ingénierie d'application. Nous proposons dans un premier temps de faire face au challenge $\mathcal{C}_1$ en *(i)* définissant des modèles de caractéristiques des fournisseurs de Cloud et en *(ii)* associant les caractéristiques à des concepts présents dans l'ontologie du Cloud. Dans un deuxième temps, nous traitons le challenge $\mathcal{C}_2$ en proposant une ontologie de propriétés non-fonctionnelles dont les concepts sont associés aux attributs des caractéristiques.

Les Utilisateurs. SALOON est destiné à toute personne souhaitant migrer une application existante dans le Cloud ou déployer un nouveau service, sous réserve d'avoir les connaissances techniques nécessaires au bon fonctionnement de l'application. Il peut donc s'agir d'un développeur, d'un architecte logiciel, d'un expert en Cloud Computing qui souhaite un support logiciel pour l'aider dans ses démarches voire même d'un fournisseur de Cloud (*e.g.*, pour tester ses services SaaS). Les modèles utilisés par le cadre logiciel (les ontologies et les MCs) sont établis et mis à jour par des experts en Cloud Computing.

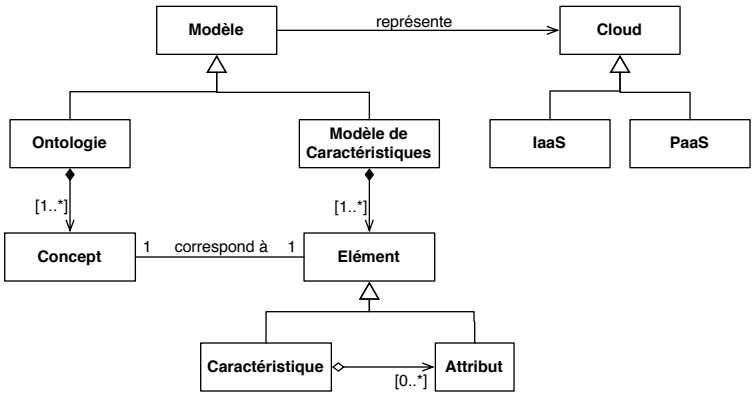
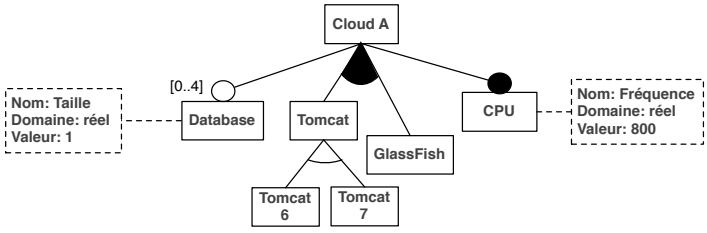


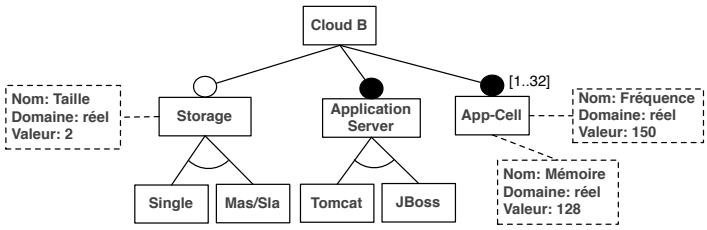
Figure 3. *Meta-modèle de SALOON (Extrait)*

Ingénierie du Domaine. L'architecture de SALOON (FIG. 3) repose sur deux parties distinctes, *(i)* les ontologies d'une part et *(ii)* les MCs d'autre part. Les ontologies permettent de formaliser la connaissance d'un domaine particulier, en l'occurrence ici du Cloud Computing. Elles servent ici également de pont sémantique entre l'utilisateur et l'outil de raisonnement du système, les MCs. En effet, certaines caractéristiques définies dans des MCs distincts peuvent différer syntaxiquement mais être sémantiquement équivalentes. L'ontologie permet de regrouper ces caractéristiques sous un même concept. Ainsi, la caractéristique `Database` du fournisseur A (FIG. 4a) et la caractéristique `Storage` du fournisseur B (FIG. 4b) sont sémantiquement équivalentes et correspondent au même concept (*e.g.*, `Stockage`) dans l'ontologie `Cloud` (FIG. 5). Nous proposons deux ontologies. La première, `Cloud`, représente les élé-

ments techniques proposés par les fournisseurs de Cloud (*e.g.*, serveur d'application utilisé, base de données supportée, etc.). La deuxième ontologie, `Dimension`, formalise, quant à elle, le dimensionnement de ces éléments techniques, (*e.g.*, la taille de la base de données souhaitée, la fréquence CPU, etc.). La séparation en deux ontologies distinctes permet une meilleure évolution de chacune d'entre elles et garantit de garder la cohérence de l'une lors de la modification de l'autre.



(a) Modèle de caractéristiques du fournisseur A



(b) Modèle de caractéristiques du fournisseur B

Figure 4. *Différents fournisseurs de Cloud et leur* MC

Les MCs forment l'ossature du cadre logiciel pour raisonner. Chaque MC représente l'offre d'un fournisseur de Cloud Computing et les possibilités de configuration associées à cette offre (FIG. 4). Le manque d'expressivité de certaines caractéristiques a été comblé en ajoutant des informations sous forme d'un attribut, qui consiste en un triplet <*nom*, *domaine*, *valeur*>, déjà envisagé dans FODA. Ce sont ces attributs qui permettent de dimensionner la configuration. Un exemple d'attribut est illustré FIG. 4a avec la définition de la taille de la base de données (`Database`), ici 1To. À ces MCs dits étendus ont aussi été ajoutées des cardinalités [CZA 05], sous la forme d'un intervalle *[n..m]* avec *n* borne minimale et *m* borne maximale de cet intervalle. Celui-ci détermine le nombre d'instances autorisées de la caractéristique dans la configuration. Ainsi, FIG. 4b, le fournisseur `Cloud B` propose de 1 à 32 `App-cells` (mini instances de calcul comprenant 128Mo de RAM et 150MHz de CPU).

Ingénierie de l'Application. Pour l'utilisateur, les ontologies sont le point d'entrée de SALOON, permettant ainsi de délimiter le domaine. En effet, l'utilisateur ne vient pas directement configurer les MCs des fournisseurs de Cloud mais sélectionne les

concepts présents dans les ontologies en fonction de la configuration de l'application à déployer. Des correspondances sont établies entre les concepts définis dans ces ontologies et les caractéristiques des MCs des fournisseurs de Cloud ou leurs attributs. Nous définissons une correspondance entre un concept et une caractéristique (ou attribut) sur la base d'un nom identique. Ainsi, si l'utilisateur choisit le concept `Stockage` dans l'ontologie, les correspondances associées aux sous-concepts de `Stockage` impliquent la sélection de la caractéristique `Database` (respectivement `Storage`) dans le MC du fournisseur A (respectivement B), comme illustré FIG. 5. La configuration établie peut ensuite être vérifiée à l'aide de solveurs SAT dans le cas de MCs binaires et de solveurs SMT pour des MCs étendus.

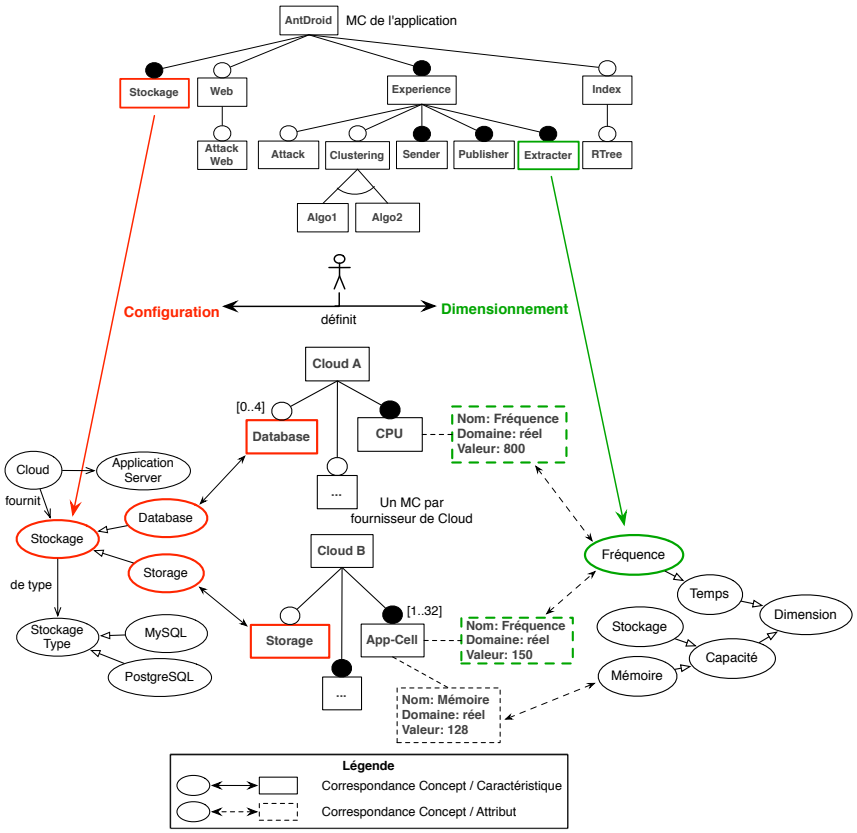


Figure 5. *Correspondances entre les ontologies et les* MCs *(Extraits).*

Réponse aux Challenges. Pour faire face au challenge $\mathcal{C}_1$ *gestion de la variabilité technique et applicative*, l'utilisateur choisit dans l'ontologie `Cloud` les éléments techniques nécessaires au bon fonctionnement de l'application en fonction de sa configu-

ration. Par le biais des correspondances entre concepts et caractéristiques, ce choix de concepts entraine la sélection des caractéristiques correspondantes dans les différents MCs et établit donc une configuration de ces MCs. La vérification par des solveurs SAT des contraintes associées à ces MCs (*e.g.*, sélection d'une caractéristique si elle est requise par une autre caractéristique, sélection de la caractéristique parent d'une caractéristique sélectionnée) permet de tester la validité des configurations sélectionnées et détermine donc les fournisseurs de Cloud capables de déployer l'application (*i.e.*, ceux dont la configuration du MC est valide). Ainsi, une application nécessitant une base de données peut être déployée sur chacun des Clouds `Cloud A` et `Cloud B` de la FIG. 4. En revanche, si cette même application doit être déployée dans un serveur d'application `JBoss`, seul le fournisseur `Cloud B` offre une configuration valide.

Plusieurs configurations de MCs peuvent être valides. Dans ce cas plusieurs fournisseurs de Cloud peuvent permettre le déploiement de l'application. Pour gérer le challenge $\mathcal{C}_2$, *gestion du dimensionnement*, l'utilisateur a alors la possibilité d'affiner la liste des fournisseurs potentiels en choisissant dans l'ontologie `Dimension` les propriétés non-fonctionnelles qu'il souhaite pour l'application à déployer (*e.g.*, la taille de l'espace de stockage, les performances CPU, etc.). Supposons qu'il veuille déployer une application `App` ayant besoin de peu de fréquence CPU (*e.g.*, moins de 800MHz) pour s'exécuter correctement. Il peut alors choisir entre un déploiement chez le fournisseur `Cloud A` ou chez le fournisseur `Cloud B` (FIG. 4). S'il exige une plus grande fréquence CPU, il doit s'orienter vers le fournisseur `Cloud B` qui permet de sélectionner plusieurs `App-Cells` et donc d'augmenter la fréquence CPU fournie. L'utilisateur a la possibilité de dimensionner toutes les caractéristiques possédant un attribut. Ainsi, si le déploiement de l'application `App`, en plus d'une fréquence CPU supérieure à 800MHz, nécessite aussi la configuration d'une base de données dont la capacité dépasse 2To, dimension seulement supportée par le fournisseur `Cloud A` qui permet jusqu'à 4To de stockage, l'utilisateur devra donc déterminer quelle est la dimension la plus importante pour le déploiement de son application `App` et choisir entre `Cloud A` et `Cloud B`. Des solveurs SMT peuvent ensuite être utilisés pour tester la validité des configurations de MCs étendus.

4. Implémentation & Retours d'Expériences

Implémentation. Le cadre logiciel SALOON repose sur différentes technologies correspondant aux deux formalismes utilisés, les ontologies et les modèles de caractéristiques. Pour construire nos ontologies, nous avons utilisé *Protégé-OWL* 4, un éditeur open-source Java permettant de créer et de manipuler des ontologies au format OWL [KNU 04]. *Protégé-OWL* est le plus connu et le plus utilisé des éditeurs d'ontologie. Il offre notamment la possibilité d'obtenir une représentation graphique de l'ontologie développée et permet d'exécuter des raisonneurs sur cette ontologie (*e.g.*, sur les descripteurs logiques). Les ontologies sont ensuite utilisées avec la librairie OWL API [HOR 11]. OWL API est une librairie Java open-source sous licence LGPL qui permet de créer et de manipuler des ontologies. C'est cette librairie qui nous permet de faire correspondre un concept de l'ontologie avec une caractéristique ou un attri-

but d'un MC. En effet, la librairie offre la possibilité de décomposer l'ontologie et d'en lister ses concepts, qui sont associés à des caractéristiques ou des attributs dans un fichier de correspondance. Nous utilisons ce fichier pour établir les configurations des MCs à partir de la liste de concepts choisis par l'utilisateur, en sélectionnant les caractéristiques correspondant à ces concepts. Les modèles de caractéristiques avec attributs et cardinalités ont été décrits avec le langage *TVL* (Textual Variability Language) [CLA 11]. Basé sur une syntaxe proche du C, il offre des mécanismes permettant à la fois d'être compréhensible et concis tout en supportant des besoins d'expressivité avancés tels que les attributs et cardinalités d'une caractéristique.

Retours d'Expériences. MC *vs ontologie.* Sur le plan technique, TVL n'en est à l'heure actuelle qu'au stade de la proposition de langage et est en cours d'implémentation. Il est cependant possible de se passer de TVL et des MCs étendus en utilisant des ontologies pour modéliser les différentes offres de Cloud Computing. L'approche est identique, seule la modélisation des fournisseurs de Cloud change. L'utilisation d'ontologies à malgré tout un certain nombre d'inconvénients. Les ontologies ne sont pas dédiées à la gestion de la variabilité et leur utilisation n'est pas adaptée lorsqu'il s'agit de gérer les caractéristiques et les contraintes associées. Pour jouer correctement le rôle d'un MC, les ontologies doivent donc être mises en relation avec un ensemble de contraintes. Nous pensons donc associer les ontologies à des contraintes *OCL* (Object Constraint Language). Pour reprendre l'exemple du dimensionnement du CPU (l'application nécessite au moins 800MHz de CPU), on peut associer une contrainte OCL spécifiant que la quantité de CPU donnée par la configuration soit supérieure à celle spécifiée par l'utilisateur lors du dimensionnement. Pour faciliter la gestion de ces contraintes, les ontologies initialement décrites en OWL peuvent être décrites en diagramme de classes *UML* (Unified Modeling Language), de nombreux outils issus de L'Ingénierie Dirigée par les Modèles permettant de manipuler le binôme UML-OCL. Il serait d'ailleurs intéressant de proposer une comparaison entre les MCs et le couple UML-OCL en termes d'expressivité, de résolution de contraintes, de temps de traitement, etc. afin de définir si le raisonnement de SALOON doit être basé sur des MCs ou des ontologies.

Évolution des ontologies et des MCs. L'utilisation exclusive d'ontologies a l'avantage de ne proposer qu'un seul formalisme de modélisation et réduit donc le coût de gestion lié aux évolutions du Cloud Computing. En effet, ces évolutions doivent être répercutées dans les différents modèles de SALOON. A l'heure actuelle, chaque MC doit être mis à jour en fonction de l'évolution de l'offre du fournisseur de Cloud. Afin de maintenir les correspondances entre MCs et ontologies cohérentes, ces dernières doivent également évoluer en conséquence.

Idéalement, et très certainement dans la suite de nos travaux, les ontologies et les MCs seront rendus publics et mis à disposition sur le Web. Ils pourront donc être consultés et mis à jour régulièrement (par le biais d'une plateforme collaborative par exemple) par des experts du Cloud Computing. Ainsi, à chaque démarrage de SALOON, la dernière version de l'ontologie et/ou des MCs pourra être téléchargée et SALOON sera donc à jour des évolutions des différents fournisseurs du Cloud Computing.

5. Travaux Connexes

Plusieurs approches utilisent des ontologies comme fondement pour trouver un fournisseur de Cloud. Dastjerdi *et al.* présentent une architecture basée sur des ontologies pour déployer des piles logicielles sur des fournisseurs de IaaS [DAS 10]. Ils proposent une ontologie qui décrit les exigences fonctionnelles et non-fonctionnelles de l'utilisateur en termes de Qualité de Service et une ontologie décrivant différents services fournis au niveau IaaS. Ils font ensuite correspondre les deux ontologies pour déterminer quel service répond le mieux aux besoins de l'utilisateur. Notre proposition va dans ce sens mais propose de chercher le fournisseur le plus approprié niveau IaaS et niveau PaaS. De plus, elle prend en compte la configuration de ces services et leur composition (*i.e.*, la possibilité de chercher un fournisseur de Cloud offrant plusieurs services). Dans [KAN 11], les auteurs présentent un système à base d'agents de découverte de services de Cloud. Ils proposent en outre des taxonomies qui sont des concepts hiérarchisés des niveaux PaaS et IaaS. Le système (*i.e.*, les agents) raisonne sur les relations entre les différents services et établit des similarités entre les requêtes de l'utilisateur et les services de Cloud pour établir un classement des services les plus pertinents. Les auteurs poursuivent donc le même but que nous mais là encore, l'utilisateur ne peut chercher qu'un service à la fois (*e.g.*, un service de Cloud offrant du CPU) et les ontologies proposées ne correspondent pas tout à fait à nos besoins.

Des approches basées sur les Lignes de Produits Logiciels et les modèles de caractéristiques sont également proposées. Ainsi, [CAV 12] *et al.* proposent une adaptation du développement basé sur les LPLs pour déployer leur système de contrôle de santé (HW). Ils incorporent dans le MC de l'application à déployer des caractéristiques "de Cloud" (*e.g.*, `Amazon S3` pour le stockage, `Google Authentication` pour le login) qu'ils ont recueillies en étudiant des applications déjà déployées sur le Cloud. Contrairement à notre démarche, ils viennent donc modifier le MC d'origine de l'application et, en quelque sorte, influent sur le choix final du fournisseur de Cloud. Ils utilisent également des MCs étendus mais l'utilisation des attributs n'est pas expliquée précisément. Dans [SCH 12], les auteurs utilisent l'ingénierie des LPLs pour configurer des applications de Cloud à plusieurs utilisateurs. Ils s'appuient sur un MC étendu pour représenter la variabilité des fonctionnalités et des qualités de services de l'application à déployer et proposent d'utiliser dans leurs travaux futurs un processus de configuration par étapes pour configurer l'application. Un expert de chaque *niveau* (IaaS, PaaS, SaaS) vient à tour de rôle configurer son niveau. Le choix du fournisseur de Cloud n'est donc pas possible et le déploiement dépendra du IaaS/PaaS configuré. D'autres auteurs ([QUI 12], [LEN 12]) présentent une approche dirigée par les modèles de caractéristiques pour créer et gérer des configurations et déployer des images virtuelles sur un IaaS. Ils considèrent une image virtuelle comme une LPL et utilisent les MCs pour représenter les configurations. Chaque caractéristique de la configuration est un logiciel à installer dans l'image virtuelle. Contrairement à notre approche, ils n'ont pas à choisir parmi les différentes plateformes de déploiement puisqu'ils configurent toute la pile logicielle qui va accueillir l'application, qui ne peut être déployée qu'au niveau IaaS.

6. Conclusion et Perspectives

Dans cet article, nous avons présenté un cadre logiciel, SALOON. Basé sur les ontologies et les MCs, il aide l'utilisateur à choisir en fonction de la configuration de l'application le fournisseur de Cloud qui permettra son déploiement. Les ontologies représentent le domaine du Cloud, tandis que les MCs décrivent la variabilité des différents fournisseurs de Cloud. L'utilisation de MCs étendus permet en outre le dimensionnement de cette variabilité. Nous utilisons SALOON comme outil de configuration et de dimensionnement de l'application à déployer sur le Cloud. Nous proposons d'utiliser les ontologies comme point d'entrée et définissons des correspondances entre les caractéristiques des MCs et les concepts des ontologies. La sélection de concepts par l'utilisateur correspondant aux exigences de l'application entraîne la définition de configurations des MCs de Cloud. Si cette configuration est valide, le fournisseur de Cloud associé au MC peut exécuter cette application, et la configuration peut ensuite être dimensionnée pour profiter de l'élasticité du Cloud. Notre approche fait face aux challenges identifiés en SEC. 2. Pour le premier challenge, à savoir le support de la variabilité technique, nous proposons une ontologie de concepts techniques communément utilisés dans le Cloud et des MCs des différents fournisseurs de Cloud. Concernant le second challenge, le support du dimensionnement, nous proposons une ontologie et des attributs associés aux caractéristiques des MCs pour décrire les dimensions des différents concepts techniques. Dans les deux cas, le cadre logiciel établit une correspondance entre les concepts des ontologies et les caractéristiques des MCs. Nous avons implémenté notre approche en utilisant des technologies connues du domaine et avons décelé quelques limitations.

Notre perspective sera dans un premier temps de lever les limitations techniques afin d'évaluer et de valider complètement notre approche. Par la suite, nous pensons que notre approche peut être étendue afin de gérer les différentes Qualités de Service souhaitées par l'utilisateur (*e.g.*, coût du déploiement d'une configuration donnée) pour proposer la meilleure solution possible parmi les solutions envisageables. De plus, il serait intéressant d'aller vers un outil d'optimisation de la configuration s'appuyant sur les différents solveurs pour aider l'utilisateur dans sa configuration.

7. Bibliographie

[BEN 05] BENAVIDES D., TRINIDAD P., RUIZ-CORTÉS A., « Automated reasoning on feature models », *Proceedings of the 17th international conference on Advanced Information Systems Engineering*, CAiSE'05, Berlin, Heidelberg, 2005, Springer-Verlag, p. 491–503.

[CAV 12] CAVALCANTE E., ALMEIDA A., BATISTA T., CACHO N., LOPES F., DELICATO F. C., SENA T., PIRES P. F., « Exploiting software product lines to develop cloud computing applications », *Proceedings of the 16th International Software Product Line Conference - Volume 2*, SPLC '12, New York, NY, USA, 2012, ACM, p. 179–187.

[CLA 11] CLASSEN A., BOUCHER Q., HEYMANS P., « A Text-based Approach to Feature Modelling : Syntax and Semantics of TVL », *Science of Computer Programming, Special*

Issue on Software Evolution, Adaptability and Variability, vol. 76, n° 12, 2011, p. 1130–1143.

[CLE 01] CLEMENTS P., NORTHROP L. M., *Software Product Lines : Practices and Patterns*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[CZA 05] CZARNECKI K., HELSEN S., EISENECKER U. W., « Formalizing cardinality-based feature models and their specialization », *Software Process : Improvement and Practice*, vol. 10, n° 1, 2005, p. 7-29.

[DAS 10] DASTJERDI A. V., TABATABAEI S. G. H., BUYYA R., « An Effective Architecture for Automated Appliance Management System Applying Ontology-Based Cloud Discovery », *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, Washington, DC, USA, 2010, IEEE Computer Society, p. 104–112.

[GRU 93] GRUBER T. R., « A translation approach to portable ontology specifications », *Knowl. Acquis.*, vol. 5, n° 2, 1993, p. 199–220, Academic Press Ltd.

[HOR 11] HORRIDGE M., BECHHOFER S., « The OWL API : A Java API for OWL ontologies », *Semantic Web*, vol. 2, n° 1, 2011, p. 11-21.

[KAN 90] KANG K. C., COHEN S. G., HESS J. A., NOVAK W. E., PETERSON A. S., « Feature-Oriented Domain Analysis (FODA) - Feasibility Study », rapport, 1990, The Software Engineering Institute.

[KAN 11] KANG J., SIM K. M., « Cloudle : an ontology-enhanced cloud service search engine », *Proceedings of the 2010 international conference on Web information systems engineering*, WISS'10, Berlin, Heidelberg, 2011, Springer-Verlag, p. 416–427.

[KNU 04] KNUBLAUCH H., FERGERSON R. W., NOY N. F., MUSEN M. A., « The Protégé OWL plugin : An open development environment for semantic web applications », Springer, 2004, p. 229–243.

[LEN 12] LE NHAN T., SUNYÉET G., JÉZÉQUEL J.-M., « A Model-Driven Approach for Virtual Machine Image Provisioning in Cloud Computing », DE PAOLI F., PIMENTEL E., ZAVATTARO G., Eds., *Service-Oriented and Cloud Computing*, vol. 7592 de *Lecture Notes in Computer Science*, p. 107-121, Springer Berlin / Heidelberg, 2012.

[MEL 09] MELL P., GRANCE T., « The NIST Definition of Cloud Computing », rapport, 2009, National Institute of Standards and Technology.

[POH 05] POHL K., BÖCKLE G., LINDEN F. J. V. D., *Software Product Line Engineering : Foundations, Principles and Techniques*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[QUI 12] QUINTON C., ROUVOY R., DUCHIEN L., « Leveraging feature models to configure virtual appliances », *Proceedings of the 2nd International Workshop on Cloud Computing Platforms*, CloudCP '12, New York, NY, USA, 2012, ACM, p. 2 :1–2 :6.

[SCH 12] SCHROETER J., MUCHA P., MUTH M., JUGEL K., LOCHAU M., « Dynamic configuration management of cloud-based applications », *Proceedings of the 16th International Software Product Line Conference - Volume 2*, SPLC '12, New York, NY, USA, 2012, ACM, p. 171–178.