# **Recherches effectuées à partir du 01/10/13**

Encadrant : Sylvain Hallé

E-mail: shalle@acm.org

Web: [http://www.leduotang.com/sylvain](https://mailetud.uqac.ca/OWA/redir.aspx?C=NN4HNHTaPkmfIPMdGFUwjXxC4GRasNBIasZkggPCvyz5OldwFDkIfCUO8U3bwD0ZdKX4KhkyCfM.&URL=http%3a%2f%2fwww.leduotang.com%2fsylvain)

Tel: +1 418 545 5011 ext. 5280

Fax: +1 418 545 5012

Le but des recherches est de trouver des outils qui permettraient de “parser” un fichier de configuration cisco et d’en obtenir une structure abstraite (en arbre, XML). Cette structure abstraite doit être indépendante de la version du routeur/switch cisco depuis lequel on récupère la configuration.

Cela fonctionne de la même manière dans le sens inverse : le programme doit être capable, à partir de la même configuration, de générer les commandes cisco spécifiques à la version du matériel.

# Découverte du projet ValidMaker : mi-octobre

Installation du logiciel

Test du parser développé en C# au sein du logiciel

Possibilités de ce parser : ajout dynamique de commandes cisco pouvant être détectées

Limites de ce parser: il faut pouvoir abstraire sans erreur les versions de commandes.

Il n’existe pas d’émulateur de routeur cisco au sein du logiciel qui permettrait de tester si les commandes entrées fonctionnent pour récupérer/modifier les paramètres.

# 

# Recherches d’autres parsers existants : réunions du mois d’octobre

Augeas (utilisé en ligne de commandes sur Linux)

...

**CiscoConfParser (Python)**

Il s’agit d’une librairie python s’important donc en une ligne. Il suffit d’y instancier un Parseur prenant en argument le texte de la configuration (CLI) et l’on obtient ensuite un objet qui contient une représentation due cette configuration. Ce parseur permet de retrouver facilement des éléments précis de la configuration.

**Avantages:**

- récupérer des paramètres par simple appel de fonction

- parse sans erreur les configurations (même complexes) Cisco

- gère les espaces et autres caractères d’échappement

- prend en compte les expressions régulières

- peut retrouver les parents ou enfants d’un noeud

**Inconvénients:**

- ne peut pas modifier les éléments d’une configuration

- ne peut pas générer du xml

- pas d’accès à la structure de stockage interne

- demande de connaître la syntaxe précise du paramètre que l’on veut obtenir ou une expression régulière qui permet de le trouver

- cette librairie est en python donc l’interfacage avec Validmaker semble limité à première vue

Problèmes des différentes versions de CISCO qui donnent une représentation abstraite différente

Yang

MIB (validmaker développé en réaction à ça qui est pas terrible)

CIM : schéma conceptuel pouvant exprimer la configuration réseau

Test case à définir.

# Consignes pour le mois de Novembre : réunion du 08/11

Si on avait 2 versions différentes de cisco et qu'on veut les récupérer dans la même structure de données abstraite et aussi faire l'opération inverse, comme ferait-on ça? -> imaginer la conception

# Problème principal

Exemple de commande cisco (page 44 dans 1rf\_bk1.pdf) : “ip domain list”

Dans la version 10.0, la syntaxe est : ip domain-list name

Dans la version 12.2, la syntaxe est : ip domain list name (sans tiret)

Le noeud dans la représentation abstraite (méta-CLI) doit être le même, que le routeur soit en version 10.0 ou 12.2. Cette différence de syntaxe ne doit rien changer à l’arbre final.

Inversement, si on a 2 noeuds dans la représentation abstraite, il se peut que ces 2 noeuds se changent en une seule commande cisco (par exemple, adresse IP et masque) :

ip address xxx.xxx.xxx.xxx [xxx.xxx.xxx.xxx]

ip address *ip address [ mask ]*

# Travail effectué du 08/11 au 13/11

Nous avons recherché une architecture et un fonctionnement du programme qui permettrait de parser un fichier de configuration CLI (CISCO) et d’en obtenir une représentation générique, comme méta-CLI.

Nous avons réalisé un schéma UML qui se sépare en 2 principales parties :

- la première est celle correspondant à l’IOS Reference (telle qu’appelée dans le logiciel ValidMaker). Elle contient l’ensemble des syntaxes des commandes et est utilisée comme référence pour le parsing des configurations Cisco. Elle est obtenue après avoir été chargée depuis un document XML correspondant à cette référence IOS. Ce document XML aura lui-même pu être généré après avoir “parsé” la documentation CISCO (ce parsing étant réalisé une fois pour toutes, ou tout du moins jusqu’à que CISCO apporte des modifications à sa documentation, suite à la sortie d’une nouvelle version).

- la seconde correspond au schéma obtenu suite au parsing d’un fichier de configuration. Les noeuds de ce schéma générique sont liés aux noeuds de l’IOS Reference. On retrouve les différentes valeurs des paramètres de la configuration dans le schéma. La génération d’un fichier XML (méta-CLI) est quasi-triviale grace à cette architecture.

Le parser de fichiers CLI existant dans ValidMaker pourrait en partie servir et s’adapter à cette architecture.

-> Est-il envisageable de modifier la structure du méta-CLI ou est-ce obligatoire de s’y conformer strictement?

Envisons le cas où nous conserverions la structure en place. Actuellement, le Méta-CLI utilisé dans Validmaker suit un schéma très précis mais toutefois générique. Il est définit dans un fichier XSD qui fixe une certaine structure.

Un Device est définit par quelques paramètres (Description, Propriétés et services) mais surtout par son arbre de configuration (ConfigTree).

Le but premier de notre parseur est de remplir cet arbre de configuration à partir du CLI. Pour celà, nous avons regardé la structure sous jacente du ConfigTree. Il se trouve que ce noeud comporte 1 à n noeuds de type DevNode.

Le noeud DevNode est quant à lui très générique. Il comporte des attributs tels que **name** le nom du noeud ou un booléen **parameter** permettant de savoir si le noeud représente un paramètre. Il contient ensuite une valeur (string), 0 à n noeuds de paramètre et enfin 0 à n enfants. Ces enfants sont eux même de type DevNode ce qui permet de redéfinir un nouveau niveau de hiérarchie.

On peut donc remarquer que rien ne définit dans la structure le format final d’un arbre de configuration et que celà laisse une grande liberté. Le problème qui se pose est la non normalisation des informations: une même information peut être référencée sous un nom différent dans deux fichiers aux versions syntaxiquement différentes. Le schéma présent n’offre aucune garantie de ce point de vue.

Le second problème est la hiérarchisation qui est non définie. De par le format des fichiers de configuration actuels Cisco, nous obtenons toujours un arbre regroupant les paramètres d’une interface comme noeuds enfant du noeud identifiant l’interface. Ainsi, on verra toujours apparaître un noeud contenant le nom de l’interface suivi de ses enfants correspondant aux commandes la décrivant. Cette hiérarchie n’est pas définit suivant des règles mais par le format du fichier source. Il y a donc un problème de généricité à ce point de vue.

La réutilisation du format existant est possible. Toutefois, le fichier généré aura une syntaxe lourde si l’on souhaite y ajouter plus de sous niveaux dans l’organisation et le regroupement des paramètres. Enfin, certains paramètre pourraient ne pas être utilisés (comme les attributs paramètres ou known du type DevNode) car le parsing utilisé diffèrerait.

-> Langage et plateforme de prédilection pour implémenter notre architecture?

Le langage permettant un parsing aisé est le python car il s’exécute facilement sur les chaînes de caractères et permet un passage sans type des arguments d’une fonction ce qui est utile pour traiter les règles d’une grammaire. On peut y générer le xml facilement à l’aide de librairies.

Le C# semble également à privilégier car nous possédons du code sur lequel nous appuyer pour parser notre configuration. Il est possible d’instancier des objets correspondant à chaque noeuds puis de les exporter vers le format XML.

# Travail effectué du 15/11 au 26/11

Nous avons recherché une conception, un modèle qui nous permettrait de représenter l’ensemble de l’architecture cisco, en créant également une structure abstraite (générique) indépendante de la version CISCO. Nous avons amélioré au fur et à mesure le diagramme UML.

## Réunion du 26/11/2013

Avec Eric

Eric travaille sur les méthodes de pointage des éléments du modèle de configuration générique (afin de détecter des configurations de réseau et des types de services)

A faire :

* Exemple concret fichier CISCO -> XML + IOSReference correspondante. Pas obligé d’utiliser le méta-CLI, on fait comme on veut.
* Faire un mini-parser XML : fonction qui recherche l’ensemble des commandes à exécuter pour modifier un paramètre. En Python de préférence car ValidMaker est en train d’être réécrit en Python (serveur web)!!
* Produire un rapport d’étape en ANGLAIS pour fin décembre avec les objectifs, ce qu’on a cherché, et ce qu’on a réussi à faire -> doit être approuvé par les chercheurs et l’entreprise. A fournir à Omar.
* Recherches coût de stockage : taille de fichier de configuration

Infos

Demande de subvention conjointe entre UQAC-UQAM : subvention fédérale payée en partie par le gouvernement et en partie par l’entreprise Ericsson pour ce projet de recherche

## Réunion du 10/12/2013

# Liens en rapport

Référence IOS de Cisco (en 3 documents):

- <http://www.cisco.com/en/US/docs/ios/12_2/ipaddr/command/reference/fipras_r.html>

- <http://www.cisco.com/en/US/docs/ios/12_2/iproute/command/reference/fiprrp_r.html>

- <http://www.cisco.com/en/US/docs/ios/12_2/ipmulti/command/reference/fiprmc_r.html>

Quelques références sur le parsing des configs IOS:

- <http://www.andrew.cmu.edu/user/yhm/publications/cisco-adapt.pdf>

- <http://www.cs.rice.edu/~eugeneng/inm08/papers/31.pdf>

Un outil qui semble parser des fichiers de config (peut-il avec IOS?): http://augeas.net/. Intéressant: transforme les configs en un \*arbre\* en mémoire, qu'on peut modifier et re-sauvegarder en format natif.

Autre librairie à explorer (en Python cette fois): <https://pypi.python.org/pypi/ciscoconfparse/0.8.3a>

XML-PI: le "Netconf" de Cisco <http://www.cisco.com/en/US/docs/ios/netmgmt/configuration/guide/nm_xmlpi_v1.html>

Trouvé sur les forums

<http://forums.whirlpool.net.au/archive/2074638>

<http://stackoverflow.com/questions/1191031/cisco-ios-show-conf-to-xml>