Command-Line Interface to AlligaTOR

Francis Hulin-Hubard

LSV / CNRS

20 décembre 2013

CosyVerif

Command-Line Interface to AlligaTOR

Stage de M1 Projet SAR prolongé

Stage de Jia-Hua XU co-encadré par Fabrice KORDON, Alban LINARD et Francis HULIN-HUBARD.

Plan

- Contexte et expression du besoin
- Architecture du logiciel
- Installation de l'outil
- Exemples d'utilisation

Rappel : Fonctionnement de CosyVerif

- Architecture client / serveur
  - le serveur : Alligator
  - LE client : Coloane
- Communications basées sur les web services
- Format d'échanges GrML

Rappel : Fonctionnement de CosyVerif

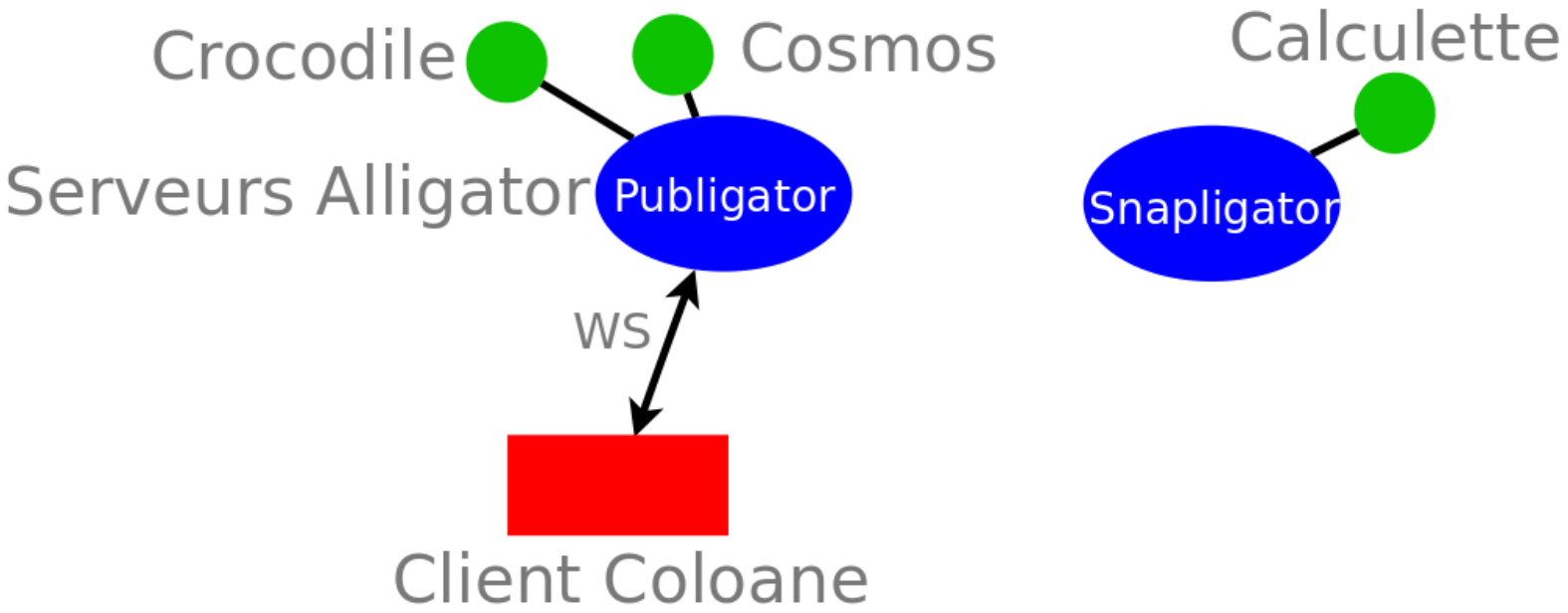


Figure : Fonctionnement de la plate-forme

Rappel : Fonctionnement de CosyVerif

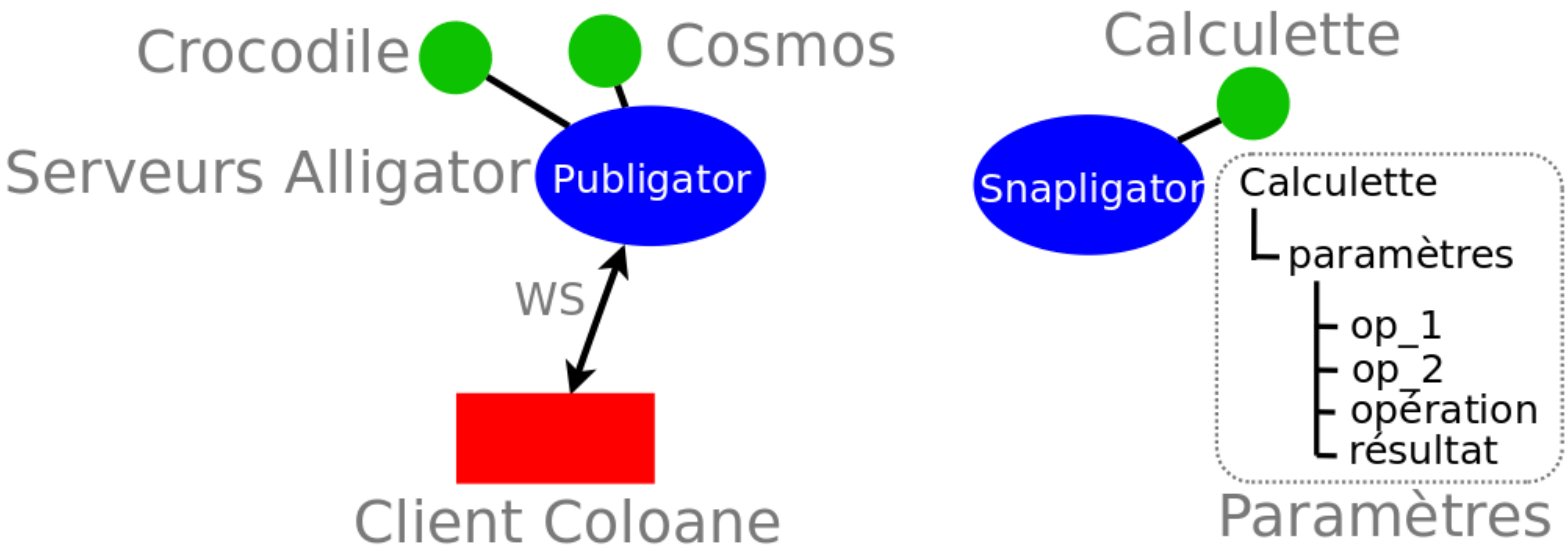


Figure : Fonctionnement de la plate-forme

Rappel : Fonctionnement de CosyVerif

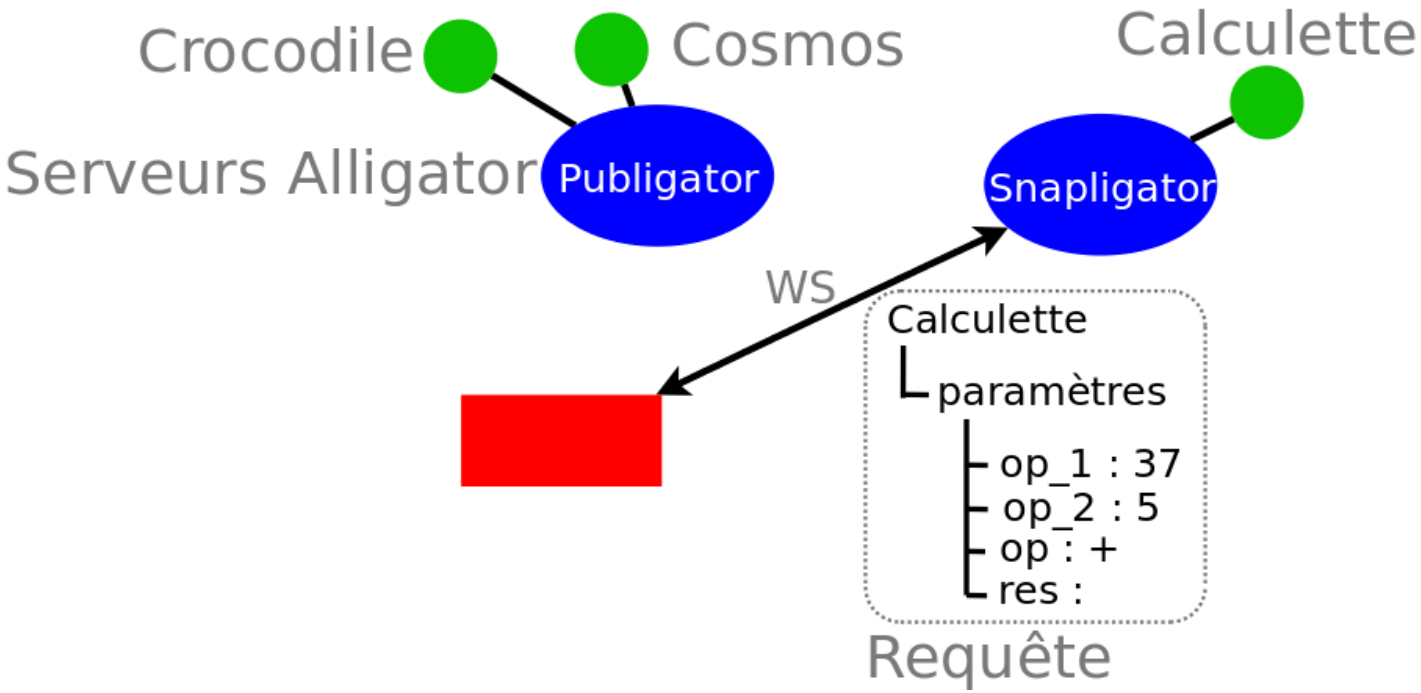


Figure : Fonctionnement de la plate-forme

Rappel : Fonctionnement de CosyVerif

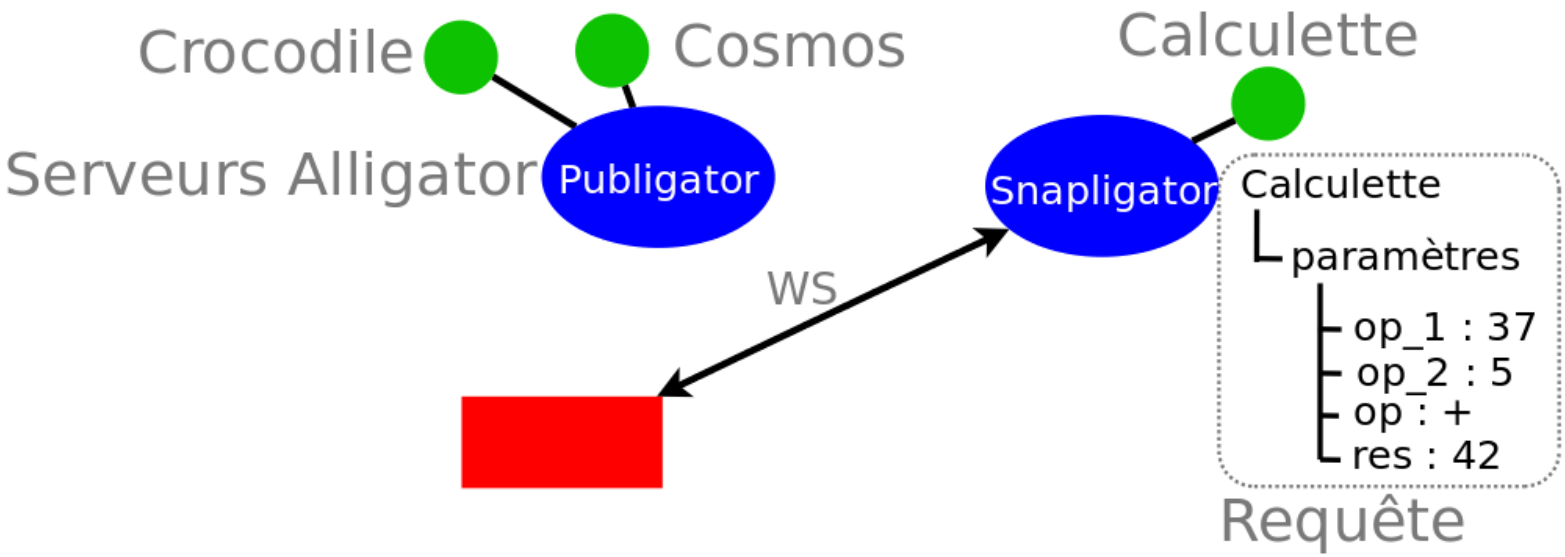


Figure : Fonctionnement de la plate-forme

Rappel : Fonctionnement de CosyVerif

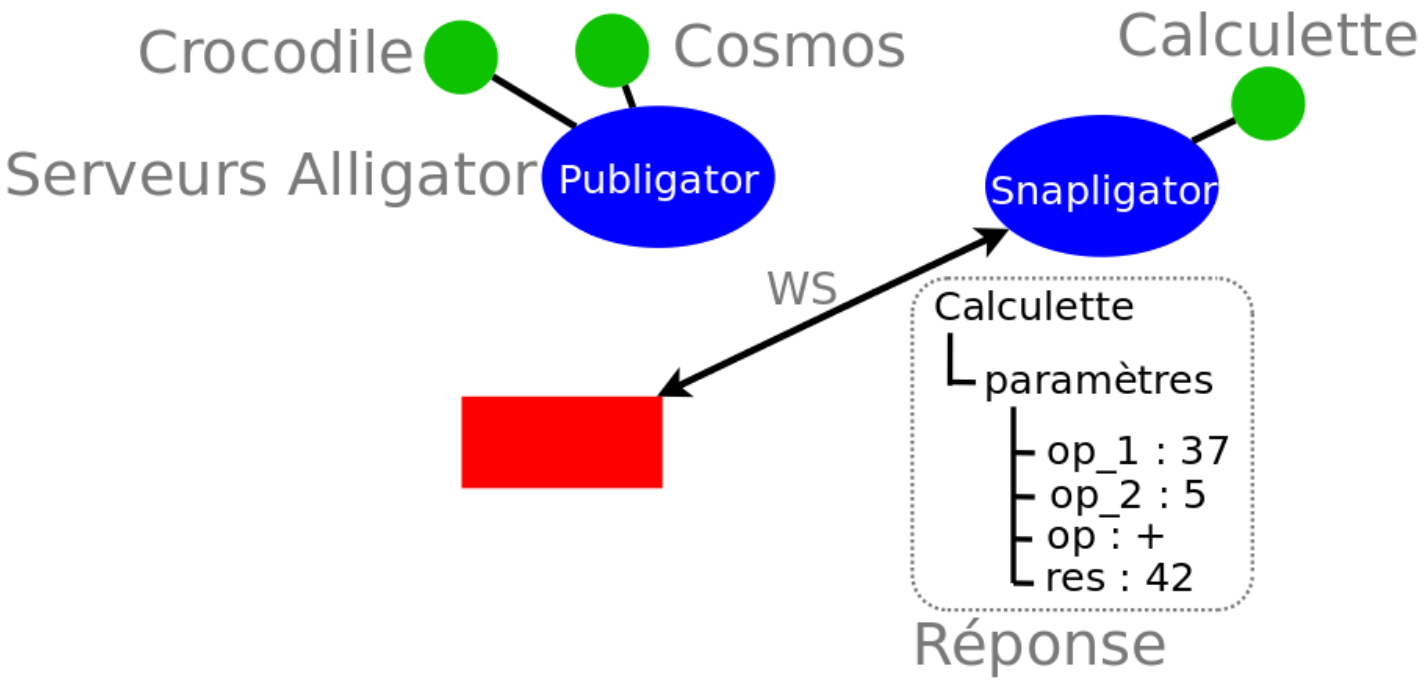


Figure : Fonctionnement de la plate-forme

Contexte et expression du besoin

La plateforme CosyVerif

- Alligator : un serveur modulaire et performant
- Coloane : un client graphique lourd

La plateforme CosyVerif

- Alligator : un serveur modulaire et performant
- **Coloane : un client graphique lourd**

Contexte et expression du besoin

La plateforme CosyVerif

- Alligator : un serveur modulaire et performant
- Coloane : un client graphique lourd

L'expérience Coloane

- Points positifs
  - multi-formalisme
  - convivial et intuitif
  - multi-plateforme

Contexte et expression du besoin

La plateforme CosyVerif

- Alligator : un serveur modulaire et performant
- Coloane : un client graphique lourd

L'expérience Coloane

- Points positifs
  - multi-formalisme
  - convivial et intuitif
  - multi-plateforme
- Points négatifs
  - ni ajout dynamique de formalisme ni lien avec FML
  - pas de support des modèles hiérarchiques
  - pas d'aspect collaboratif
  - architecture en plugins difficile à maintenir
  - multi-plateforme **mais** conservation d'une variante par architecture
  - non scriptable

Contexte et expression du besoin

Un nouveau client pour CosyVerif

- multi-formalisme
- multi-plateforme
- convivial et intuitif (autant que possible)

Contexte et expression du besoin

Un nouveau client pour CosyVerif

- multi-formalisme
- multi-plateforme
- convivial et intuitif (autant que possible)

Mais aussi :

- ajout dynamique des formalismes
- simple à maintenir (en prévision d'FMLv2)
- code réutilisable

Contexte et expression du besoin

Un nouveau client pour CosyVerif

- multi-formalisme
- multi-plateforme
- convivial et intuitif (autant que possible)

Mais aussi :

- ajout dynamique des formalismes
- simple à maintenir (en prévision d'FMLv2)
- code réutilisable

Scriptable !

Architecture du logiciel

- Structuré autour d'une bibliothèque
- Données de l'utilisateur sous une arborescence de fichiers
- Configurable par fichier de configuration

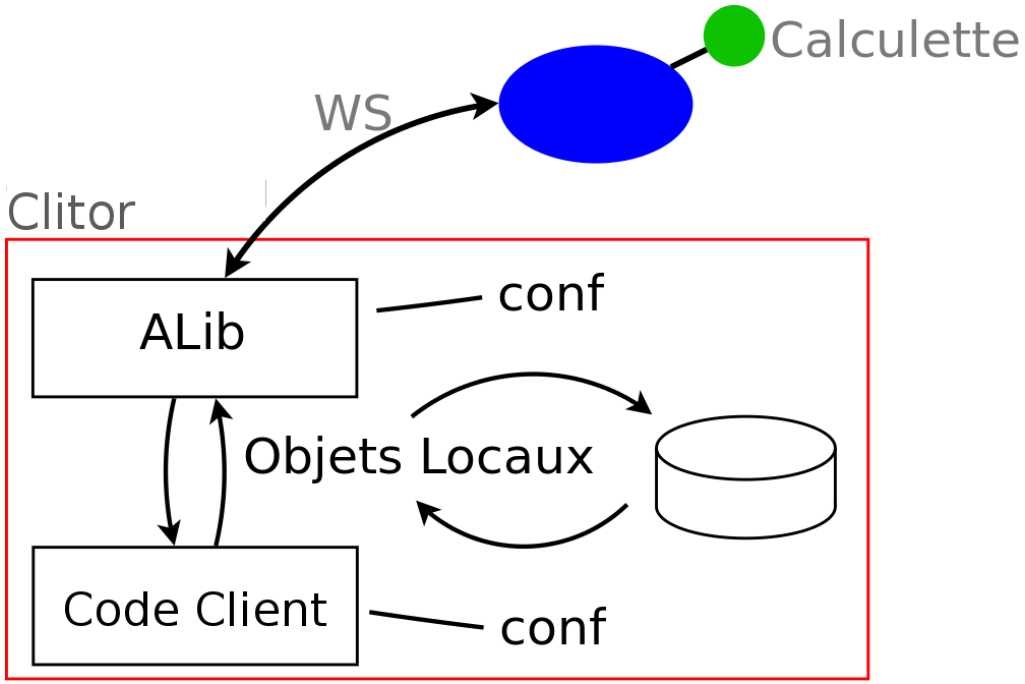


Figure : Architecture du client

Architecture du logiciel

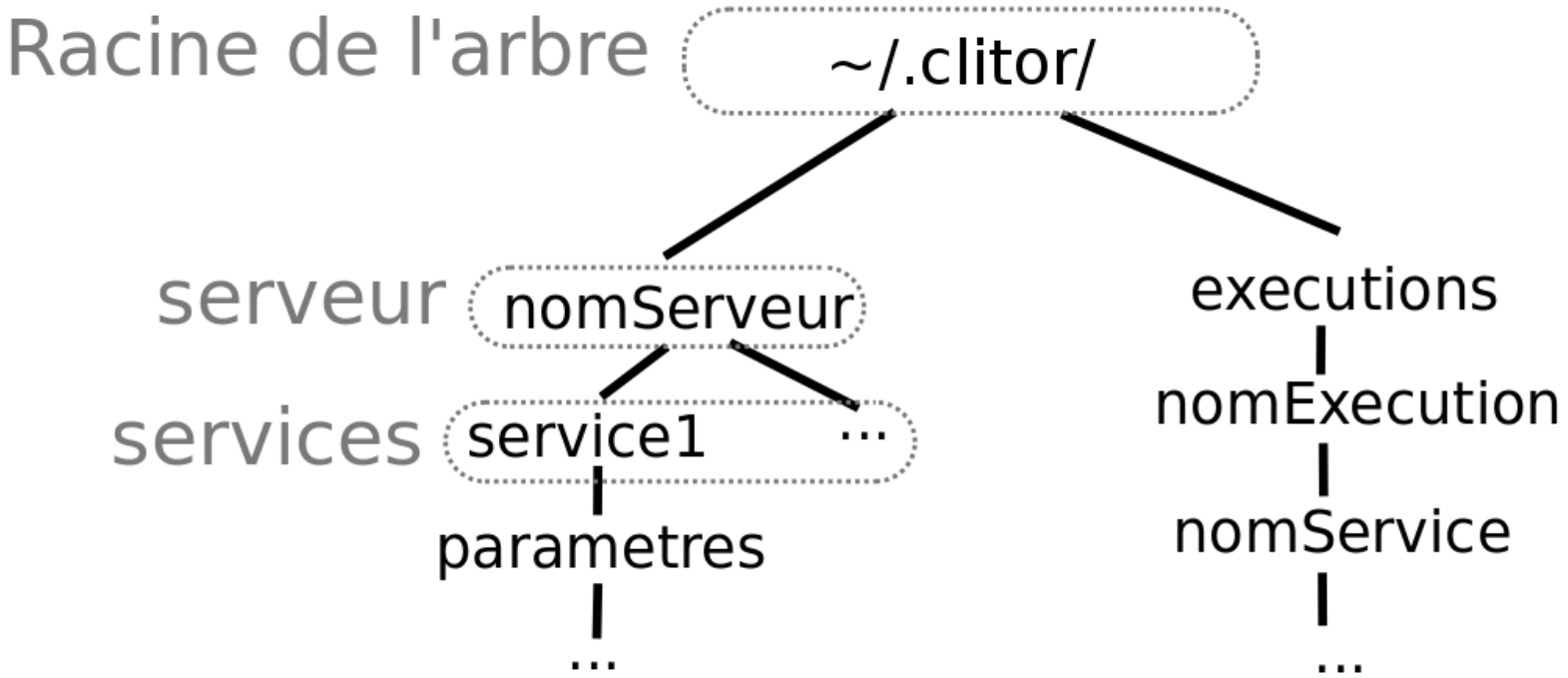


Figure : Stockage des données

Architecture du logiciel

- Un script léger écrit en PHP (objet)
- Distribué sous la forme d'un binaire «phar» d'1,6 Mo
- Multi plate-forme
- Utilisable en ligne de commande
- Simple d'approche

Installation de l'outil

Prérequis :

- PHP (5 ou supérieur) pour la ligne de commande : «php_cli»
- Activation du module «phar» (dans «php.ini»)

```
extension=phar.so
```

- Configuration d'openbasedir (dans «php.ini»)

```
; open_basedir, if set, limits all
; file operations to the defined directory
; open_basedir = /srv/http/:/home/
```

Installation de l'outil

Installation :

- Récupération des sources depuis le dépôt

```
svn export https://forge.../trunk clitor
```

- Assemblage de l'application

```
./create-phar.sh
```

- Test de l'application

```
./clitor
```

Exemples d'utilisation

- Exemple simple : la calculatrice
- Scripting d'exemple : les philosophes de 5 à 15

Exemple simple d'utilisation

Terminal

$

Données

/home/login

Exemple simple d'utilisation

Terminal

```
$ clitor services -r
snapligator :
      1) calculette
$
```

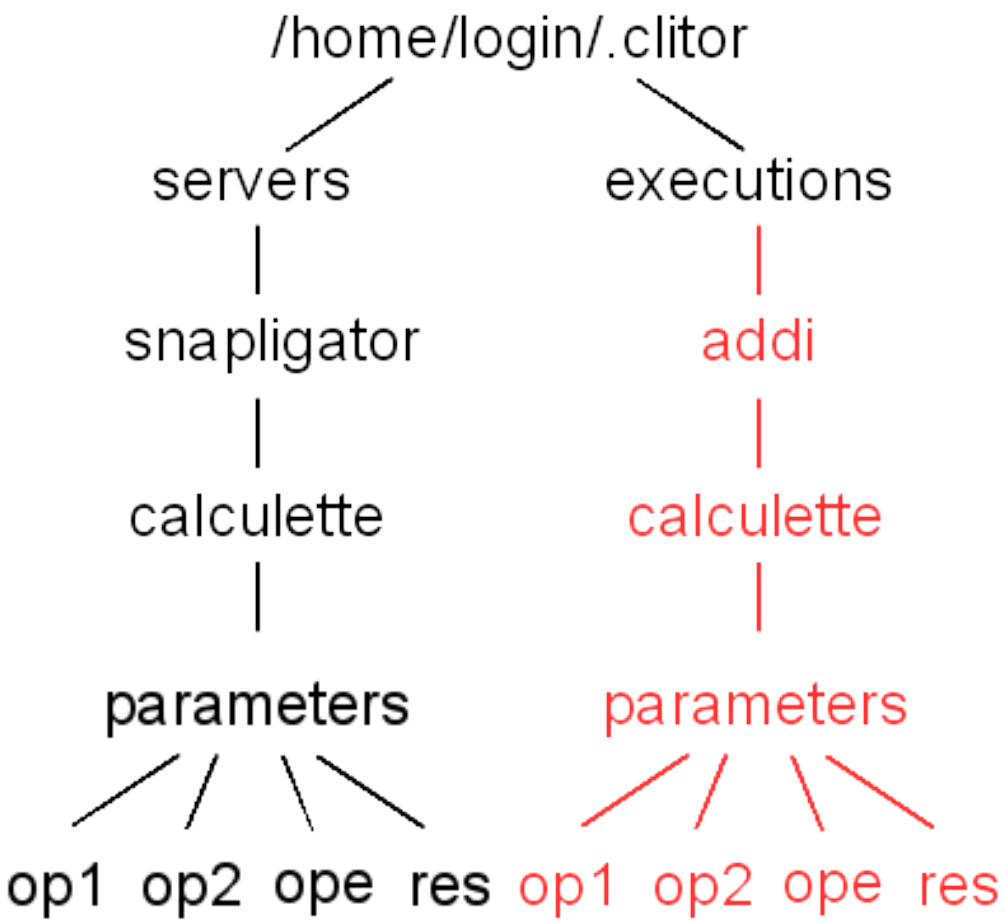
Données

Exemple simple d'utilisation

Terminal

```
$ clitor services -r
snapligator :
      1) calculette
$ clitor prepare calculette addi
$
```

Données

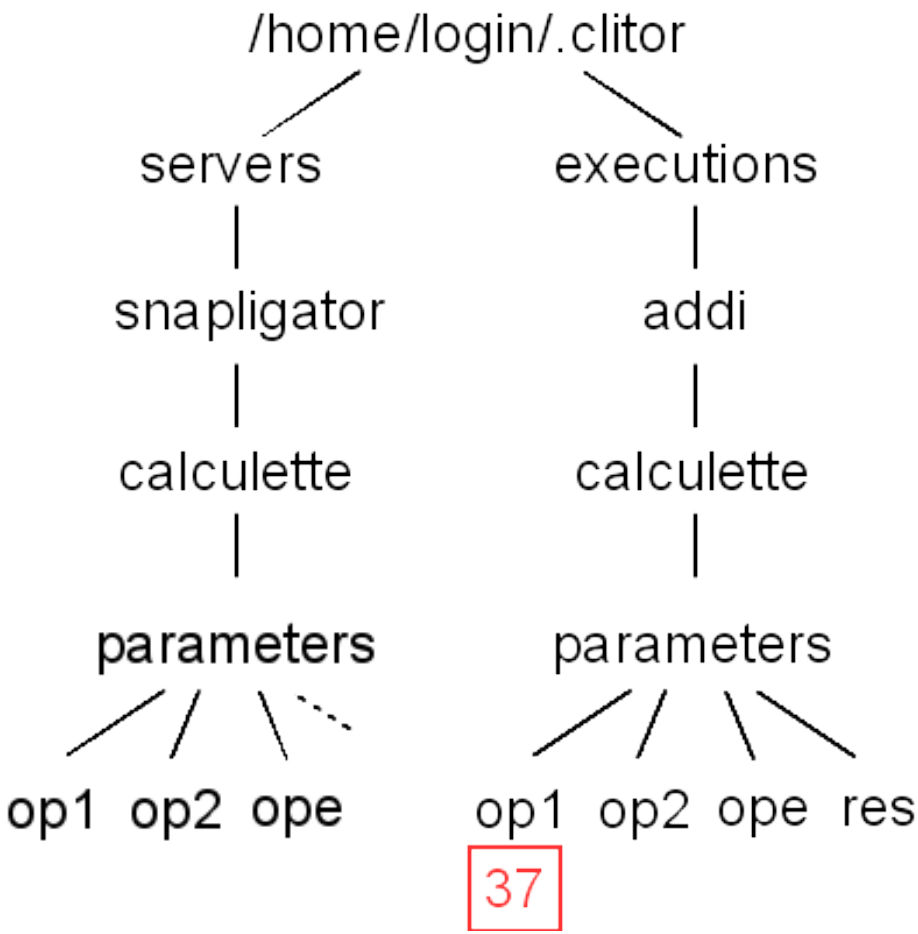


Exemple simple d'utilisation

Terminal

```
$ clitor services -r
snapligator :
      1) calculette
$ clitor prepare calculette addi
$ clitor set addi.op1 37
$
```

Données

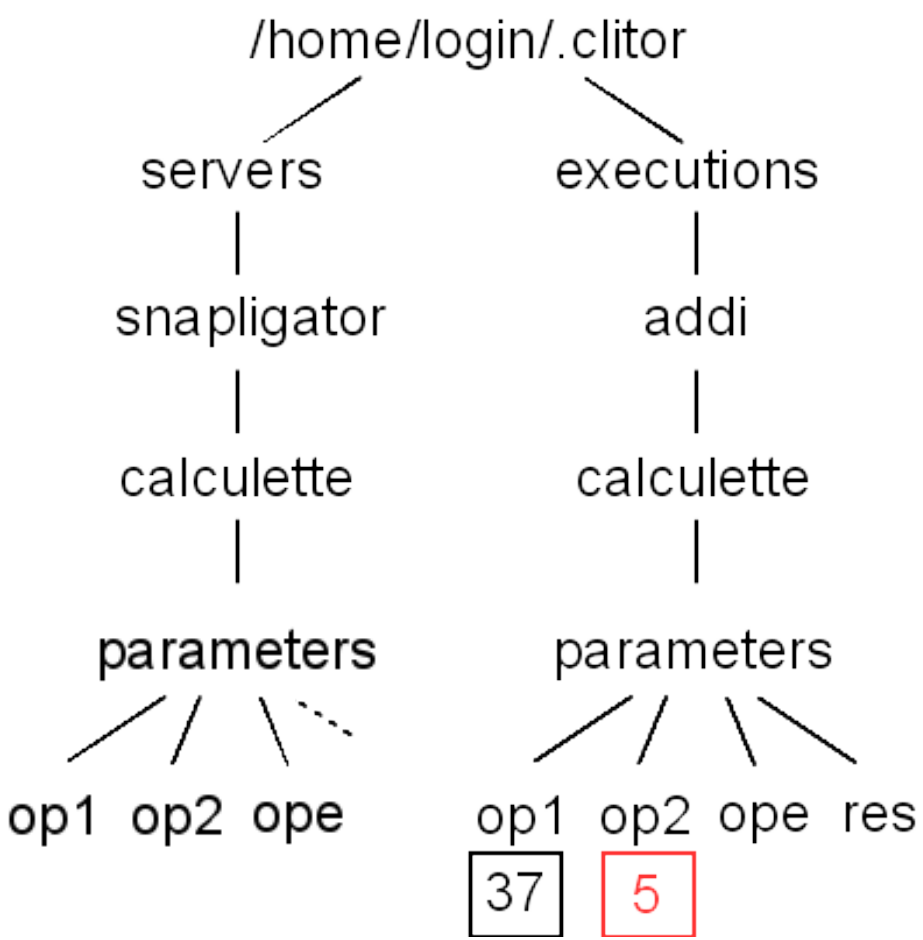


Exemple simple d'utilisation

Terminal

```
$ clitor services -r
snapligator :
      1) calculette
$ clitor prepare calculette addi
$ clitor set addi.op1 37
$ clitor set addi.op2 5
$
```

Données

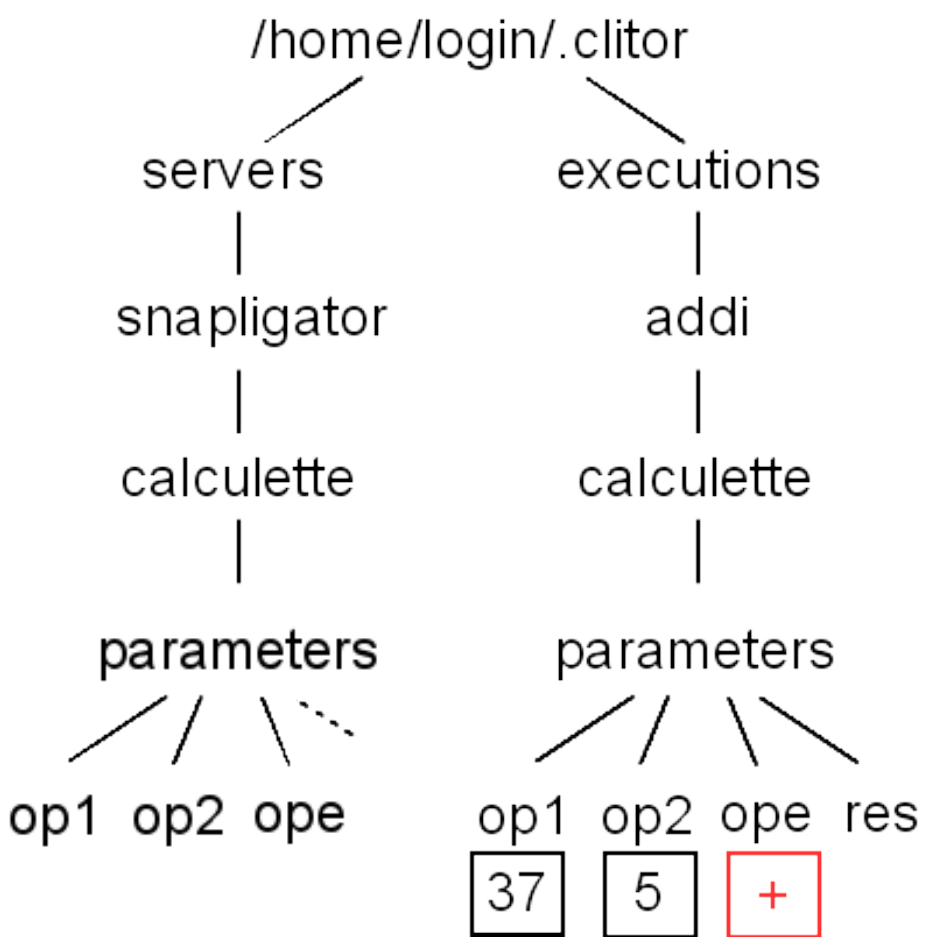


Exemple simple d'utilisation

Terminal

```
$ clitor services -r
snapligator :
      1) calculette
$ clitor prepare calculette addi
$ clitor set addi.op1 37
$ clitor set addi.op2 5
$ clitor set addi.ope +
$
```

Données

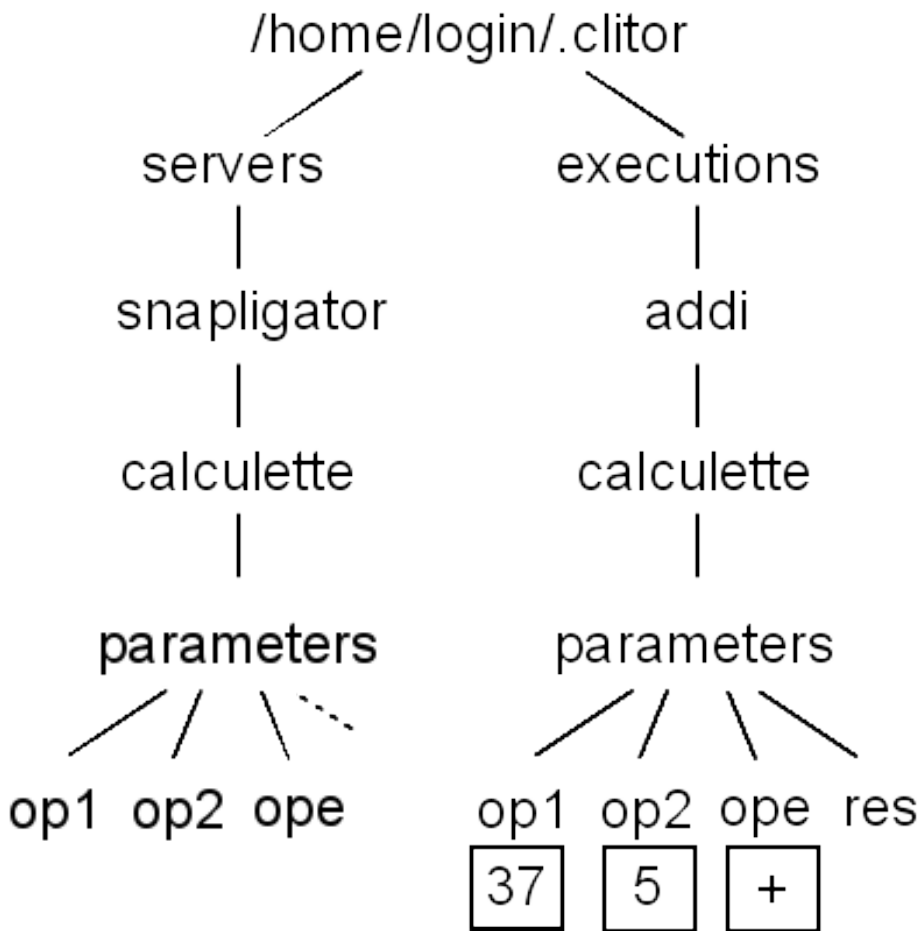


Exemple simple d'utilisation

Terminal

```
$ clitor services -r
snapligator :
      1) calculette
$ clitor prepare calculette addi
$ clitor set addi.op1 37
$ clitor set addi.op2 5
$ clitor set addi.ope +
$ clitor run -s addi
```

Données

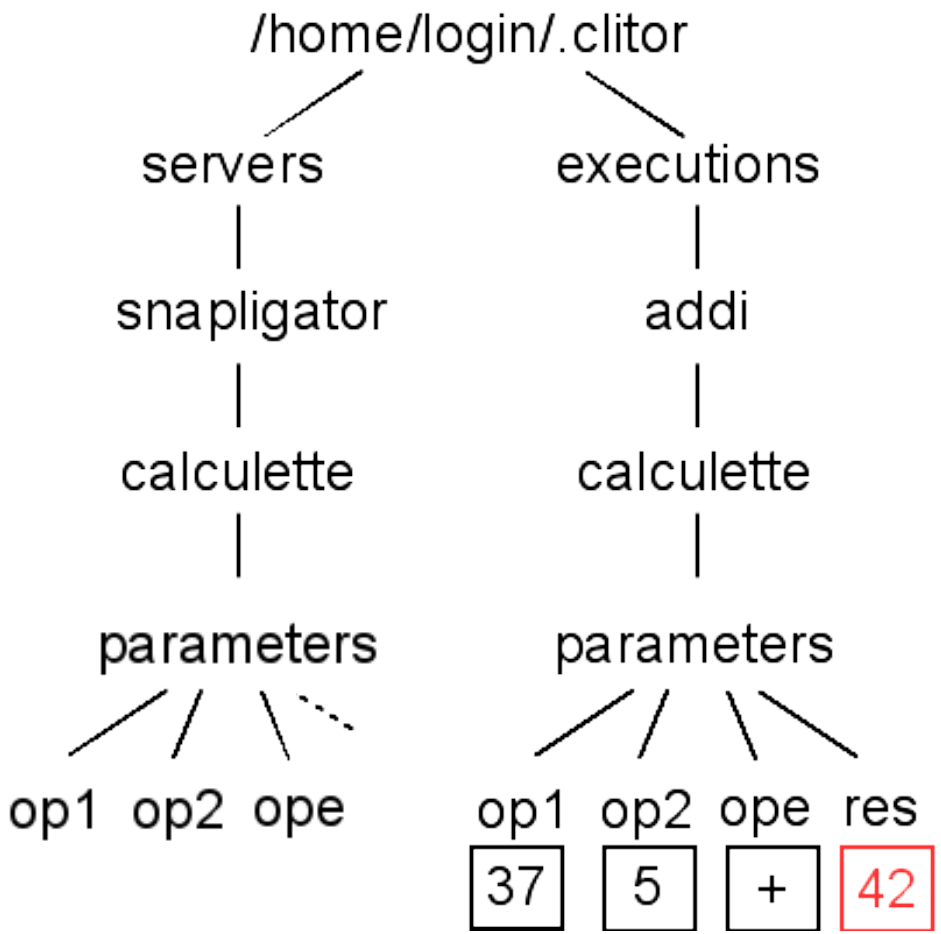


Exemple simple d'utilisation

Terminal

```
$ clitor services -r
snapligator :
      1) calculette
$ clitor prepare calculette addi
$ clitor set addi.op1 37
$ clitor set addi.op2 5
$ clitor set addi.ope +
$ clitor run -s addi
$
```

Données

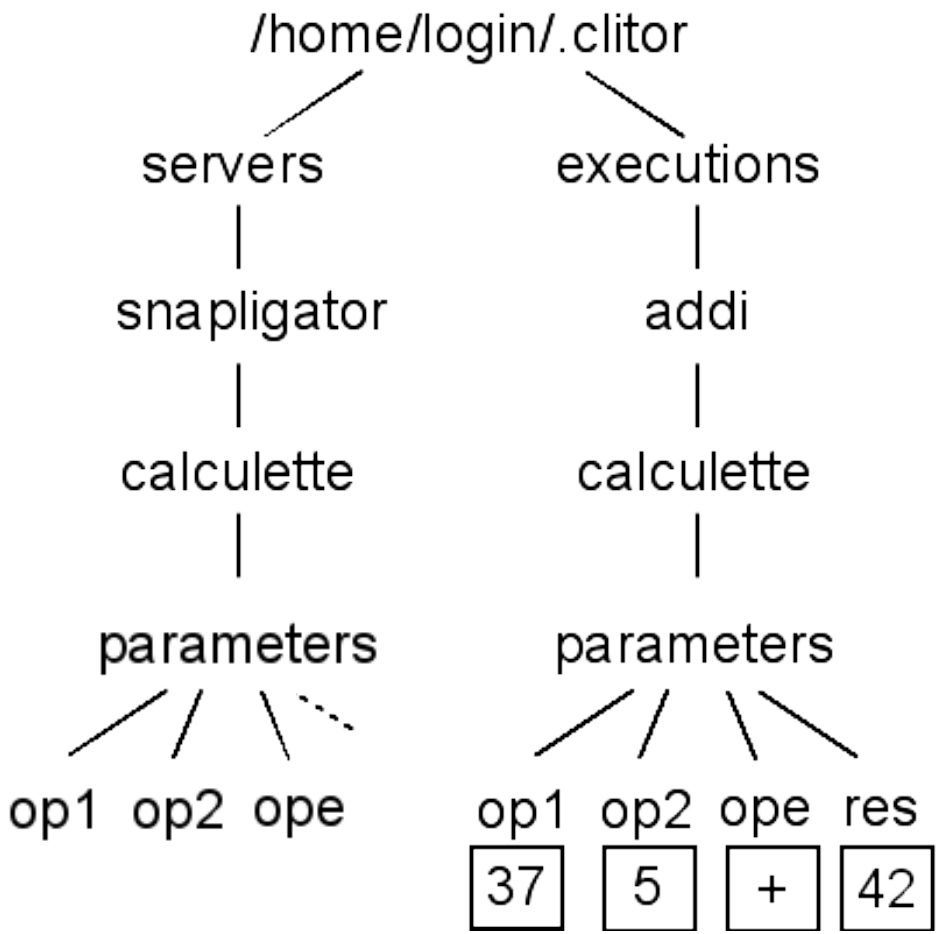


Exemple simple d'utilisation

Terminal

```
$ clitor services -r
snapligator :
      1) calculette
$ clitor prepare calculette addi
$ clitor set addi.op1 37
$ clitor set addi.op2 5
$ clitor set addi.ope +
$ clitor run -s addi
$ clitor get addi.res
42
```

Données



Exemple de scripts

Objectifs

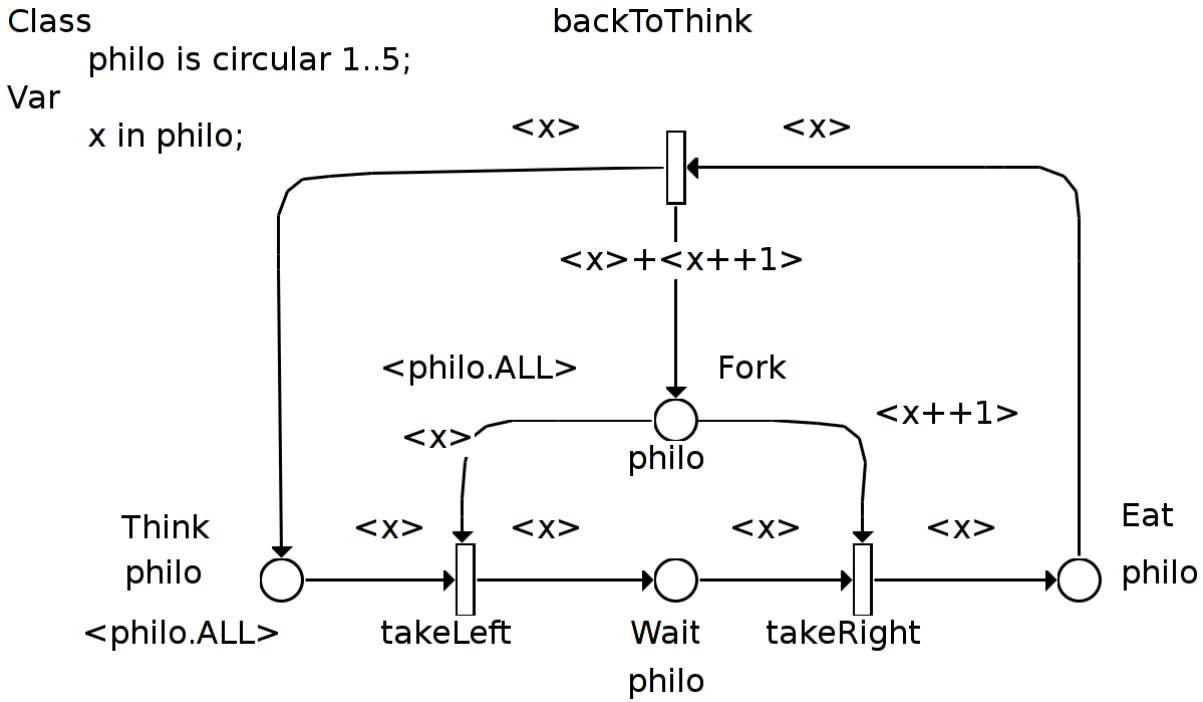
Appliquer l'outil "test" au modèle des philosophes d'ordre 5 à 15.

Exemple de scripts

Objectifs

Appliquer l'outil "test" au modèle des philosophes d'ordre 5 à 15.

- Générer un modèle au format GrML



Exemple de scripts

Objectifs

Appliquer l'outil "test" au modèle des philosophes d'ordre 5 à 15.

- Générer un modèle au format GrML

```
−<model id="1" formalismUrl="http://formalisms.cosyverif.org/s-net.fml">
 −<attribute name="declaration">
  −<attribute name="classDeclaration">
     <attribute name="name"> philo </attribute>
   −<attribute name="classType">
    −<attribute name="classIntInterval">
       <attribute name="lowerBound"> 1 </attribute>
       <attribute name="higherBound"> 5 </attribute>
     </attribute>
    </attribute>
    <attribute name="circular"> true </attribute>
   </attribute>
  −<attribute name="variableDeclaration">
    <attribute name="name">x</attribute>
    <attribute name="type">philo</attribute>
   </attribute>
 </attribute>
```

Exemple de scripts

Objectifs

Appliquer l'outil "test" au modèle des philosophes d'ordre 5 à 15.

- Générer un modèle au format GrML
- Modifier le modèle : baliser les variables

<__ORDRE__>

Exemple de scripts

Objectifs

Appliquer l'outil "test" au modèle des philosophes d'ordre 5 à 15.

- Générer un modèle au format GrML
- Modifier le modèle : baliser les variables
- Scripter l'invocation

```
$ for i in {5..15} ; do
> echo "> Lancement ordre $i" ;
> clitor prepare test philo$i ;
> clitor set philo$i.model "$(sed -e "s/<__ORDRE__>/$i/g" \
> < philo.grml)" ;
> clitor run philo$i ;
> done
```

Conclusion

- Une brique importante du futur de CosyVerif
- Un outil adapté pour le développeur et les benchs
- Un futur éditeur textuel de modèles