

Point sur COSMOS DSL

Denis Conan et Léon Lim



Réunion Broccoli, Sophia Antipolis

Septembre 2009

Sommaire

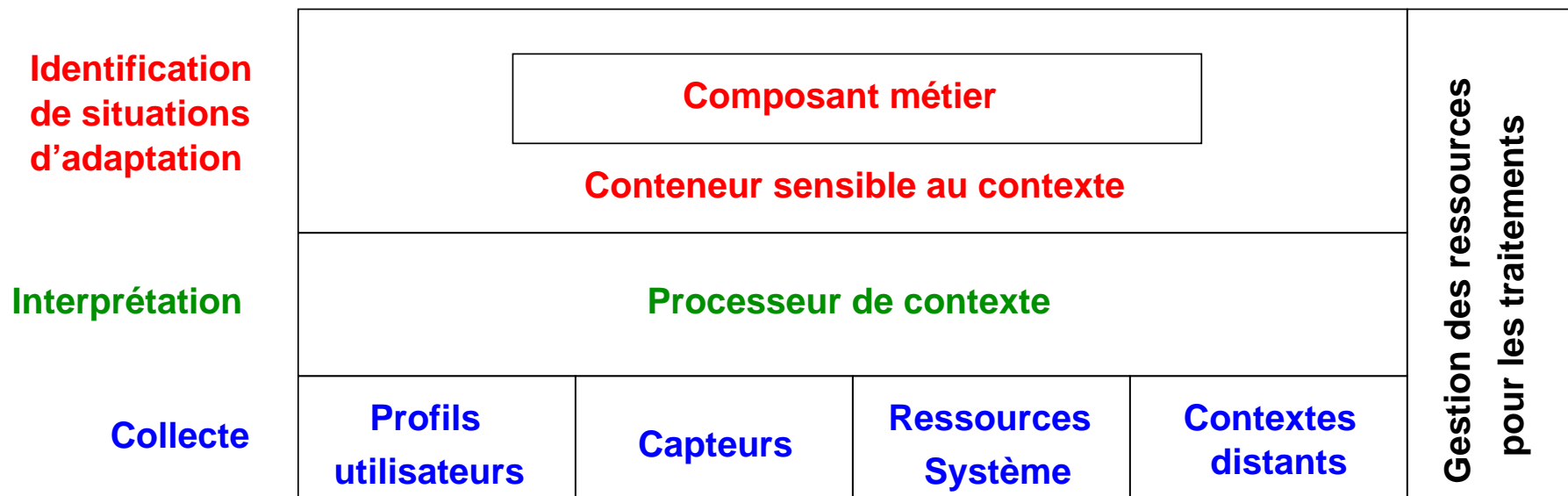
1	Contexte de l'étude	3
2	État de l'art des langages de collecte et de composition	8
3	Processus de développement du DSL COSMOS.....	10
4	Analyse de domaine avec modélisation FODA.....	11
5	Conception	13
6	Évaluation, discussion	18
7	État d'avancement et travaux futurs	19

1 Contexte de l'étude

1.1	Architecture de COSMOS	4
1.2	Concepts de base de COSMOS.....	5
1.3	Processus de conception manuelle avec COSMOS.....	6
1.4	Motivations pour une nouvelle approche.....	7

1.1 Architecture de COSMOS

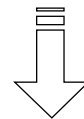
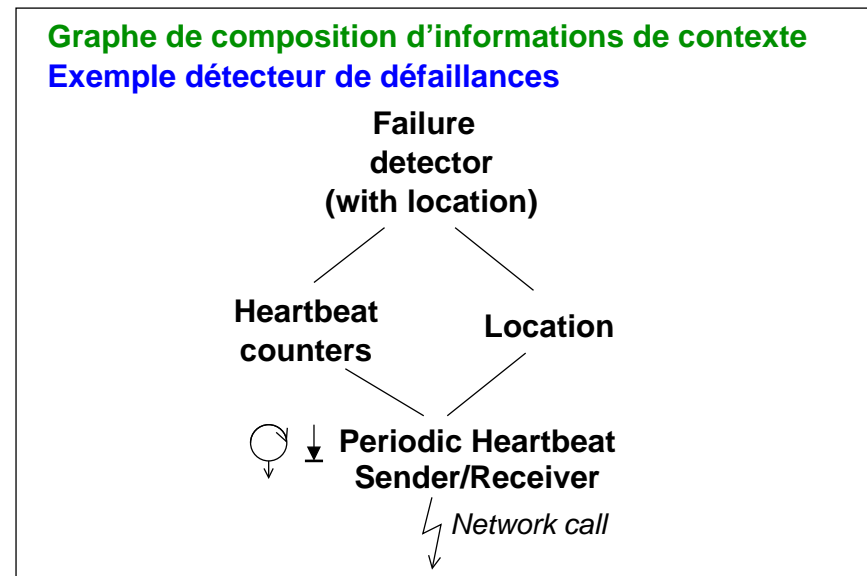
- Séparation des préoccupations / fonctionnalités
- Flexibilité grâce à une architecture à base de composants
 - ◆ Conception utilisant les principes de base des intergiciels
 - ▶ Modèle de composants : FRACTAL [Bruneton et al., 2006]
 - ▶ ADL : FRACTAL ADL [Leclercq et al., 2007]



1.2 Concepts de base de COSMOS

- **Nœud de contexte** : information de contexte modélisée par un composant logiciel
 - ◆ Définition inspirée de celle du composant [Szyperski, 2002] :
 - ▶ un nœud de contexte est une **unité de composition** avec des **interfaces serveurs** contractuellement spécifiées et des dépendances **explicites** vers d'autres nœuds de contexte
 - ▶ un nœud de contexte peut être déployé **indépendamment** et est sujet à des **compositions** par des tiers
- **Politique de gestion de contexte** : hiérarchie avec partage de nœuds de contexte
 - ◆ Définition inspirée de celle de l'architecture logicielle [Bass et al., 1998] :
 - ▶ une politique de gestion de contexte est une partie de la structure ou la structure d'un **graphe de nœuds contexte** qui inclut les nœuds de contexte, les **propriétés** de ces nœuds et les **relations** entre ces nœuds

1.3 Processus de conception manuelle avec COSMOS



Traduction manuelle

Architecture écrite en Fractal ADL

Description de la composition d'informations de contexte

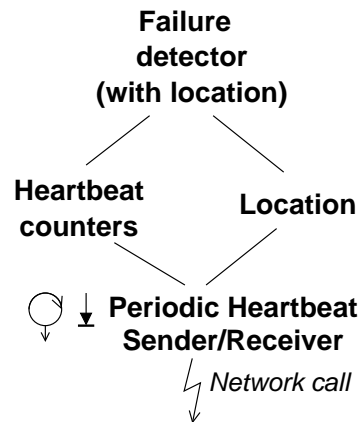
```

<definition name="FailureDetector" extends="ContexteNode">
  <component name="operator" definition="FailureDetectorCO(resourceName=>FailureDetector)"/>
  <attributes signature="cosmos.ContextOperatorAttributes">
    <attribute name="isactiveobserver" value="true"/>
    <attribute name="periodnotify" value="500"/>
  </attributes>
  <component name="child1" definition="PeriodicHeartbeatCounters"/>
  <component name="child2" definition="PeriodicHeartbeatLocation">
    <component name="child" definition="HeartbeatMgr">
      <component name="rm" definition"./child1/child/rm"/>
    </component>
  </component>
</definition>
  
```

1.4 Motivations pour une nouvelle approche

- FRACTAL ADL : des notations techniques difficiles à appréhender
- Langage dédié : notations déclaratives plus spécifiques [Mernik et al., 2005]

Graphe de composition d'informations de contexte Exemple de détecteur de défaillances



Description de la composition d'informations de contexte

```

sensor HeartbeatMgr=PeriodHeartbeatRM[BO,AO] ;
processor PeriodicHeartbeatCounter=PeriodicHeartCounterCO(HeartbeatMgr.extract
(hb-counters-chunk));
processor PeriodicHeartbeatLocation=PeriodicHeartbeatLocationCO(HearbeatMgr.extract
(location-chunk));
processor FailureDetector=FailureDetectorCO[ressource => FailureDetector]
(PeriodicHeartbeatCounters,PeriodicHeartbeatLocation);
  
```

Composer les informations de contexte en utilisant des idiomes

Traduction automatique avec traçabilité des idiomes
vers les patrons d'architecture

Architecture écrite en Fractal ADL

Description de la composition d'informations de contexte

```

<definition name="FailureDetector" extends="ContexteNode">
  <component name="operator" definition="FailureDetectorCO(resourceName=>FailureDetector)"/>
  <attributes signature="cosmos.ContextOperatorAttributes">
    <attribute name="isactiveobserver" value="true"/> <attribute name="periodnotify" value="500"/>
  </attributes>
  <component name="child1" definition="PeriodicHeartbeatCounters"/>
  <component name="child2" definition="PeriodicHeartbeatLocation">
    <component name="child" definition="HeartbeatMgr"> <component name="rm" definition=".child1/child/rm"/> </component>
  </component>
</definition>
  
```

2 État de l'art des langages de collecte et de composition

■ Critères d'évaluation d'un DSL (en anglais, *Domain Specific Language*)

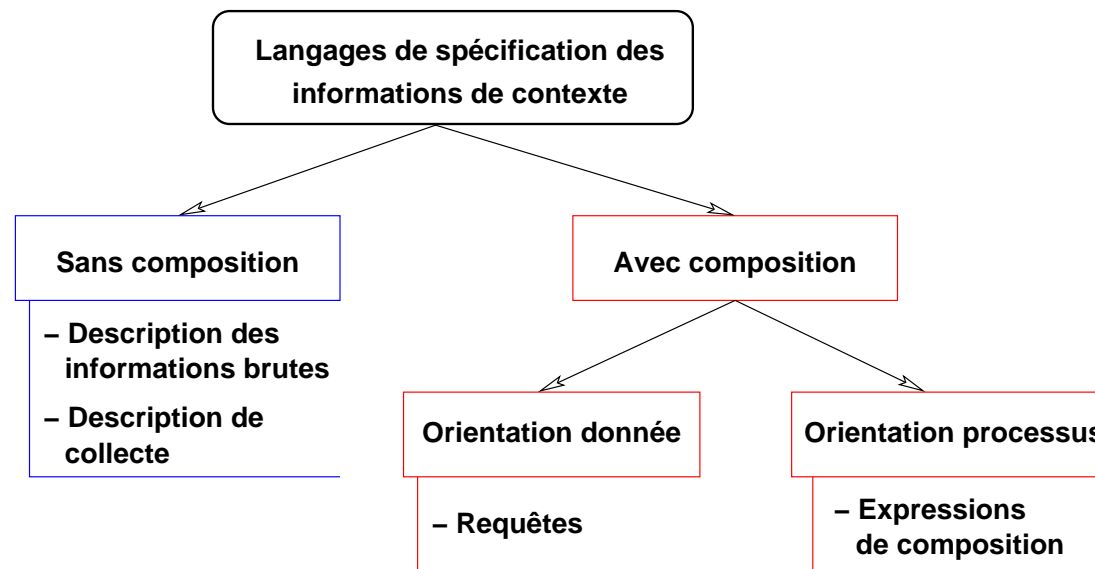
◆ Critères généraux [Mernik et al., 2005]

- ▶ Facilité d'utilisation et concision
- ▶ Extensibilité pour la réutilisation
- ▶ Validation, testabilité, optimisation

◆ Critères spécifiques au domaine

- ▶ Composabilité
- ▶ Partage

■ Classification des langages de gestion d'informations de contexte

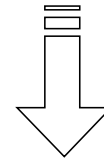
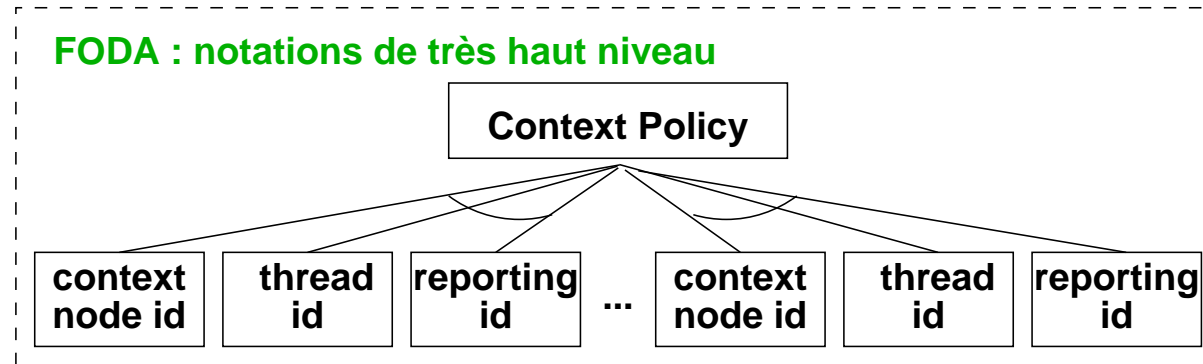


2.1 Tableau récapitulatif

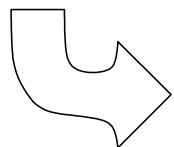
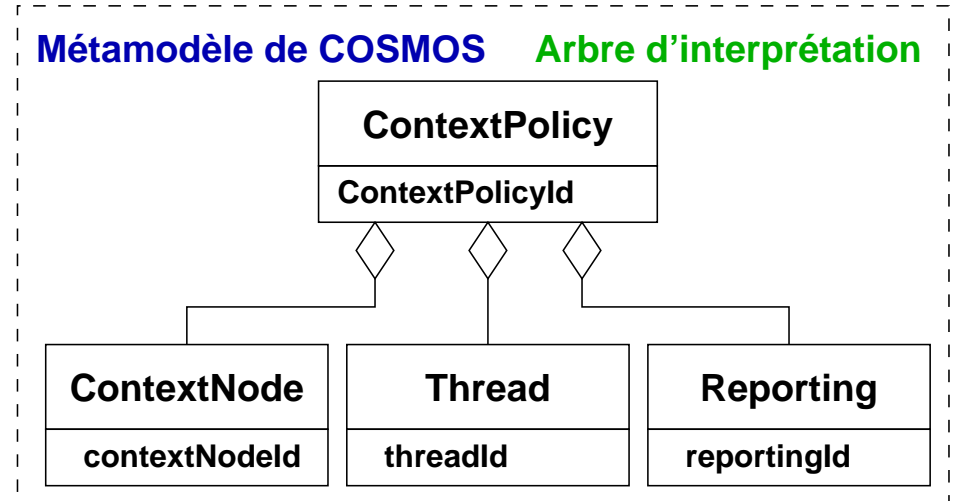
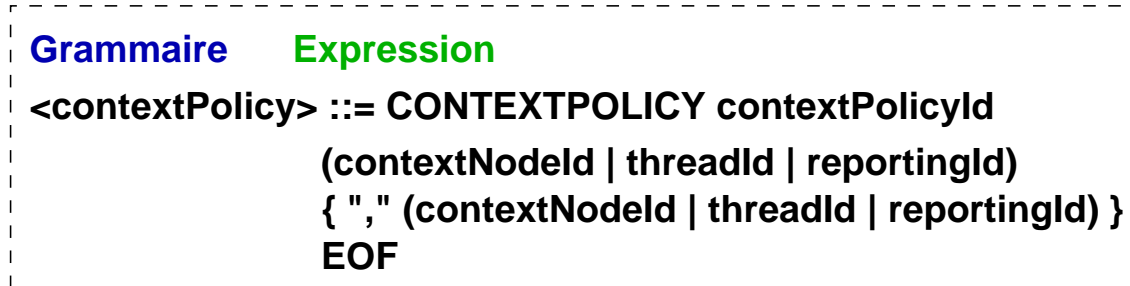
Orientation	Description		Donnée					Processus		
	SensorML	RDF	Orientés SQL	QBE	CQP	iQL	Phoenix	Gaia	SAFRAN	QML
Simplicité d'utilisation	-	#	+	+	#	#	+	+	+	#
Extensibilité		+			+			-		
Analysabilité		#		+	+	#	#	#		
Sûreté			#				-	-	#	#
Réutilisation	#	#			#	#		#		
Optimisation					#					
Testabilité			#					#		
Composabilité			#	#			#	#	#	#
Partage	#							#		

3 Processus de développement du DSL COSMOS

ANALYSE



CONCEPTION

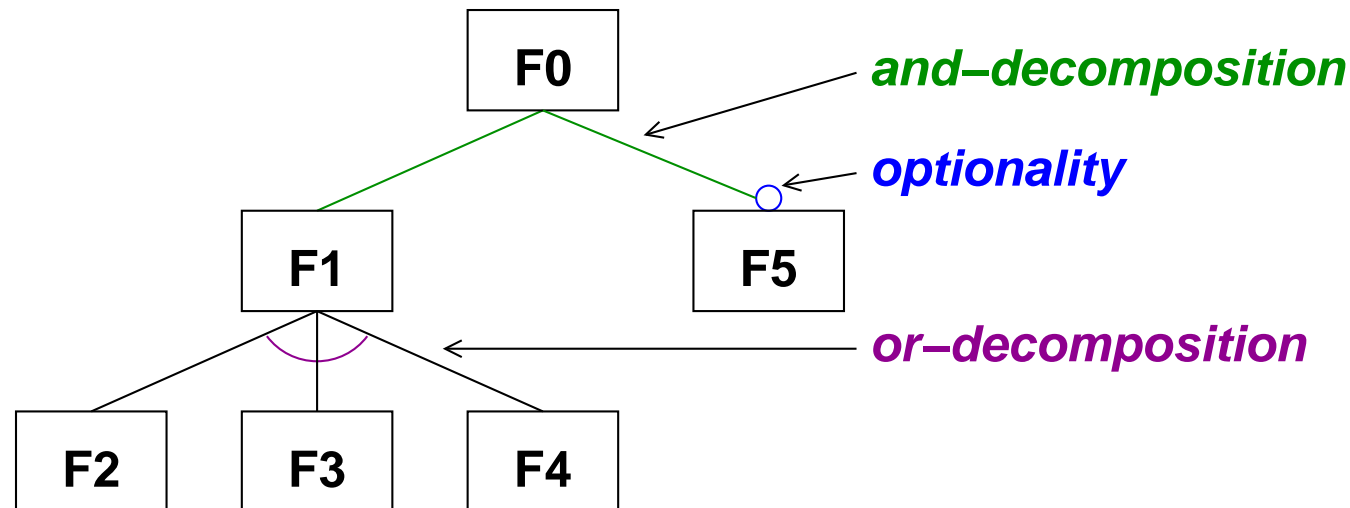


Arbre de syntaxe



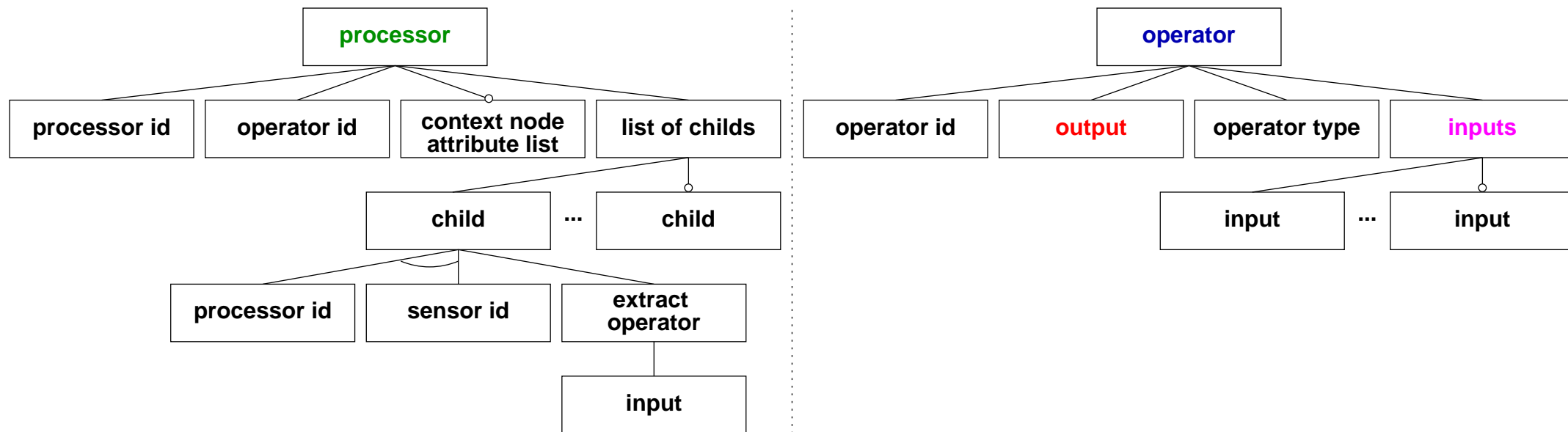
4 Analyse de domaine avec modélisation FODA

- *Feature Oriented Domain Analysis* [Kang et al., 1990]
 - ◆ Une des méthodologies d'analyse préconisée par [Mernik et al., 2005]
 - ◆ Dans notre cas, les *features* sont les *concepts du langage*
- Notation graphique



4.1 Exemple des concepts processeur et opérateur

- **Processeur** = identifiant + opérateur + attribut + nœuds enfants
- **Opérateur** = Identifiant + rapport de contexte en **sortie** + type (p.ex. classe Java)
+ rapport de contexte en **entrée**
 - ◆ Cas particulier de l'opération d'extraction de morceaux de messages
 - ▶ Utilisation très fréquente \implies construction spécifique



5 Conception

5.1	Extrait de la grammaire : type LL(2)	14
5.2	Analyse sémantique	17

5.1 Extrait de la grammaire : type LL(2)

```

contextPolicy ::= 'POLICY' contextPolicyId '=' ident {',' ident}
contextNode  ::= 'NODE' ident '=' contextNodeDef ';'
contextNodeDef ::= ((ident | '(' operatorDef ')')
                    ('[' attributeCNList ']')? '(' childList ')')
                    | resourceMngrDef
operator      ::= 'OPERATOR' operatorID '=' operatorDef ';'
operatorDef   ::= (ident | '{' messageDef '}') className
                ('[' attributeAssignList ']')? ((' messageDefList ')')?
resourceMngrDef ::= (ident | '{' messageDef '}') className
                ('[' attributeAssignList ']')?
configuration ::= ident '[' attributeCNList ']' ';'
childList     ::= child {',' child}
child         ::= (nodeId | '(' contextNodeDef ')')
                ('.' 'EXTRACT' ('[' attributeCNList ']')?
                 '(' messageDef ')')

```

```
message      ::= 'MESSAGE' messageId '=' messageDef ';'
messageDefList ::= messageId | '{' messageDef '}'
              { messageId | ',' '{' messageDef '}' }
messageDef   ::= chunkId | messageId | messageDef
              {' ,' (chunkId | messageId | messageDef)}
chunk        ::= 'CHUNK' chunkID '=' chunkClass ';'

attributeCNList ::= attributeCN {' ,' attributeCN}
attributeCN     ::= 'BO' | 'AO' | 'AN' | 'BN' | attributeAssign
attributeAssignList ::= attributeAssign {' ,' attributeAssign}
attributeAssign  ::= attributeID '=' attributeValue
```

5.1.1 Exemples d'expressions

```

CHUNK chk0 = cosmos.dsl.visitor.DummyChunk; CHUNK chk1 = c.d.v.DummyChunk;;
CHUNK chk2 = c.d.v.DummyChunk; CHUNK chk3 = c.d.v.DummyChunk;;
CHUNK chk4 = c.d.v.DummyChunk;;
MESSAGE msg0 = chk0; MESSAGE msg1 = msg0, chk1; MESSAGE msg2 = msg1, chk2;
OPERATOR op1 = msg1 opclass1[attr1 = value1]({msg2, {chk3}});
OPERATOR op2 = msg1 opclass2[attr2 = value2]({msg2});
OPERATOR op3 = {chk4} opclass3[attr6 = value6]({msg0});
NODE snsr1 = {msg2, {chk3}} rmclassname1[A0,attr3=value3];
NODE proc1 = op1[A0,BN](snsr1);
NODE proc2 = (msg1 opclass1[attr4 = value4]({msg2, {chk3}}))[A0,B0](snsr1);
NODE proc3 = op2[B0](snsr1.EXTRACT[A0,BN](msg1));
NODE proc4 = op3[B0]((op2[A0,B0](snsr1.EXTRACT[B0](msg2))).EXTRACT[A0,BN](msg0));
NODE proc5 = ({chk4}opclass3[attr6=value6](msg0))[B0]
  (((msg1 opclass2[attr2 = value2]({msg2}))
    [A0,B0]((({msg2, {chk3}} rmclassname1[A0,attr3=value3])
      .EXTRACT[B0](msg2))))
    .EXTRACT[A0,BN](msg0));

```


6 Évaluation, discussion

- **Facilité d'utilisation**
 - ◆ Langage conçu suite à une analyse de domaine
 - ◆ Idiomes correspondant aux patrons architecturaux des architectures des politiques de contexte (p.ex. hiérarchie avec partage)
 - ◆ Modularité des expressions
- **Extensibilité** : déclaration modulaire, syntaxe EBNF
 - ◆ À venir, espace de nommage et importation de fichiers existants
- **Analysabilité** : typage des éléments, idiomes
- **Réutilisation systématique** : idiomes pour le partage
- **Sûreté** : typage des éléments
- **Composabilité** : concepts opérateur et entrée/sortie
- **Partage** : partage des informations de contexte à plusieurs niveaux
- **Éléments à évaluer avec une implantation complète**
 - ◆ Testabilité
 - ◆ Optimisation

7 État d'avancement et travaux futurs

■ État d'avancement du prototype

- ◆ Limitation à la partie fonctionnelle pour l'instant
- ◆ Outils de développement
 - ▶ Projet picoforge cosmos, dépôt Subversion
 - ▶ Projet Maven, tests unitaires avec JUnit
- ◆ Langage spécifié en JavaCC, tests unitaires effectués
- ◆ Méta-modèle Cosmos implanté, tests unitaires effectués
- ◆ Transformation arbre de syntaxe concret en AST implantée, tests effectués

■ Travaux en cours et futurs

- ◆ Analyse sémantique :
 - ▶ **Contraintes Cosmos : substituabilité, etc.**
 - ★ Module cosmos-operator non finalisé
 - ▶ **Génération d'architecture FRACTAL ADL**

Références

- [Bass et al., 1998] Bass, L., Clements, P., and Kazman, R. (1998). *Software architecture in practice*. Addison-Wesley.
- [Bruneton et al., 2006] Bruneton, É., Coupaye, T., Leclercq, M., Quéma, V., and Stefani, J.-B. (2006). The Fractal Component Model and Its Support in Java. *Software—Practice and Experience, special issue on Experiences with Auto-adaptive and Reconfigurable Systems*, 36(11) :1257–1284.
- [Kang et al., 1990] Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania (USA).
- [Leclercq et al., 2007] Leclercq, M., Özcan, A., Quéma, V., and Stefani, J.-B. (2007). Supporting Heterogeneous Architecture Descriptions in an Extensible Toolset. In *Proc. 29th ACM International Conference on Software Engineering*, (USA).
- [Mernik et al., 2005] Mernik, M., Heering, J., and Sloane, A. (2005). When and How to Develop Domain-Specific Languages. *ACM Computing Surveys*, 37(4) :316–344.
- [Szyperski, 2002] Szyperski, C. (2002). *Component Software : Beyond Object-Oriented Programming, 2nd edition*. Addison-Wesley.