Langage dédié pour COSMOS

Denis Conan et Léon Lim

Présentation
Juin 2008

Sommaire

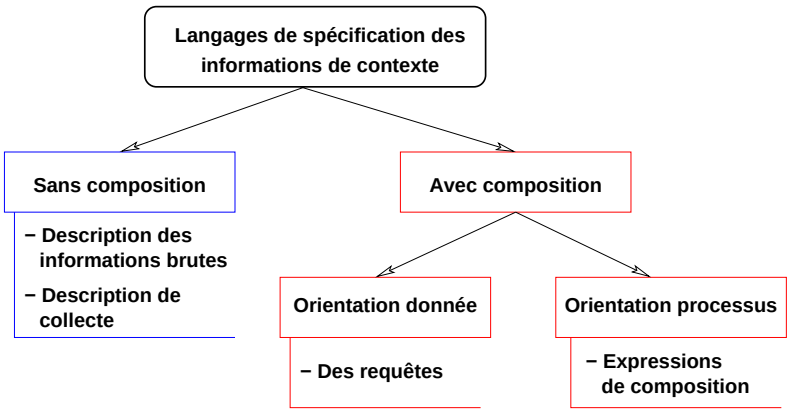
1 Décision, motivations pour un langage dédié

- FRACTAL ADL *versus* langage dédié
  - FRACTAL ADL : des notations assez techniques pas faciles à appréhender
  - Langage dédié : notations déclaratives plus spécifiques [Mernik and Sloane, 2005]
- Exigences du nouveau langage
  - Langage permettant la composition aisée des informations de contexte
    - Définition déclarative de la composition
  - Réutilisation systématique guidée par la construction du langage
  - Langage simple car expertise du domaine
    - Moins expressif qu'un langage généraliste : abstractions/notations appropriées
  - Langage optimisé car limité au domaine
  - Langage permettant la vérification de propriétés
    - Sémantique restreinte pour rendre certaines propriétés décidables
    - Séparation des concepts pour permettre la vérification de cohérence

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

2.1 Critères d'évaluation et classification des langages de gestion des informations de contexte

- Critères généraux
  - Facilité d'utilisation et concision
  - Extensibilité pour la réutilisation
  - Validation, testabilité, optimisation
- Critères spécifiques au domaine : Composabilité



2.2 Langages de description de la collecte

- Représentation des informations de contexte venant de diverses sources
- Pas de composition des informations de contexte
- Exemples
  - SensorML [Botts and Robin, 2007]
  - Langages de la famille RDF
    - CC/CP [Klyne et al., 2007]
    - UAprof [Open Mobile Alliance, 2001]
    - WURFL [Parsons, 2001]

2.3 Langages de composition orientés donnée

- Gestion des informations de contexte dans une base de données
- Langage de requête pour la composition d'informations : sélection, jointure, comptage, etc.
- Simplicité du paradigme de programmation, mais peu/pas d'expression des aspects extra-fonctionnels
- Requêtes continues : appels périodiques
- Exemples
  - CQL (*Context Query Language*) [Haghighi et al., 2006]
  - Phoenix [Boutros Saab et al., 2002]

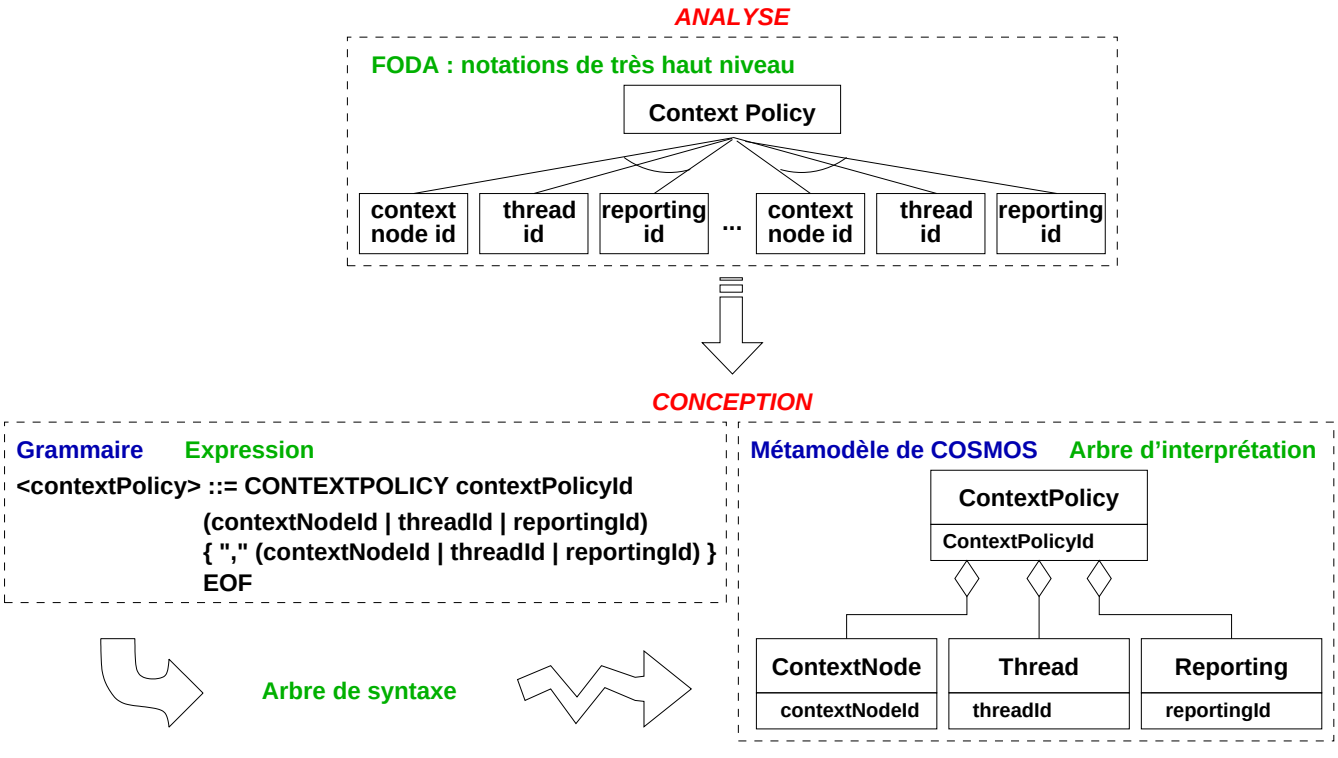
2.4 Langages de composition orientés processus

- Gestion des informations de contexte dans des unités séparées
  - « Objet », « processus », « composant » ou « service »
- Graphe (forêt) comme paradigme de construction de la composition
  - Composabilité explicite, mais vue globale est complexe
- Recherche d'informations = recherche d'un nœud dans un graphe
- Exemple
  - QML [Frølund and Koistinen, 1998] (1 seul niveau)
  - Gaia [Román et al., 2002]
  - WildCAT [David and Ledoux, 2005]

2.5 Tableau récapitulatif

	SensorML	RDF	Orientés SQL	Graphiques (QBE)	Orientés RDF (CQP)	Interactif (iQL)	Phoenix	Gaia	SAFRAN	QML
Simplicité d'utilisation	−	#	+	+	#	#	+	+	+	#
Extensibilité		+			+			−		
Analysabilité		#		+	+	#	#	#		
Sûreté			#				−	−	#	#
Réutilisation	#	#			#	#		#		
Optimisation					#					
Testabilité			#					#		
Composabilité			#	#			#	#	#	#

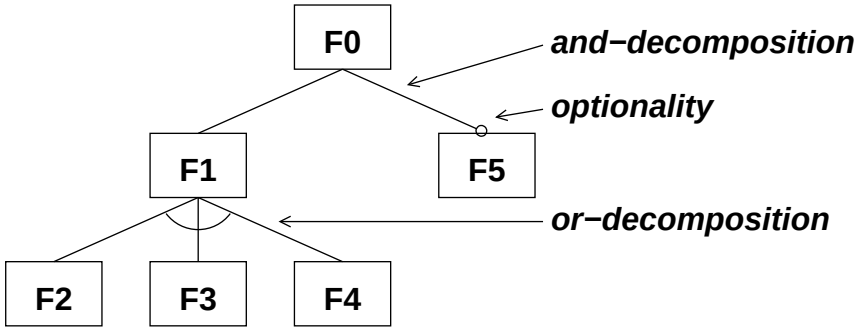
3 Processus de développement du DSL COSMOS



4 Analyse de domaine avec modélisation FODA

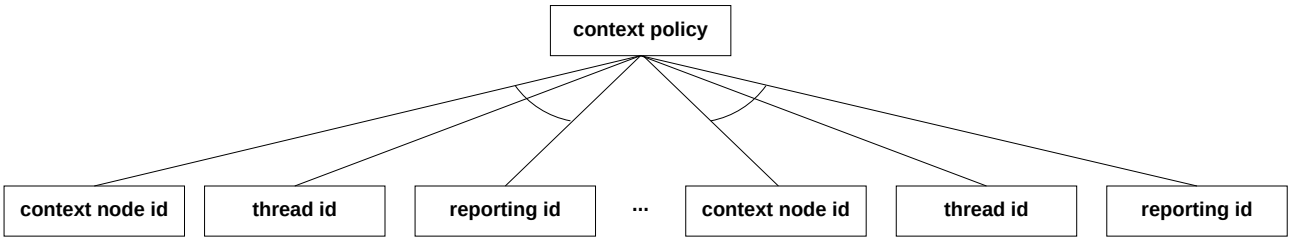
4.1 FODA

- *Feature Oriented Domain Analysis* [Kang et al., 1990, Bontemps et al., 2004]
  - Une des méthodologies d'analyse préconisée par [Mernik and Sloane, 2005]
  - Dans notre cas, les *features* sont les concepts du langage
- Notation graphique



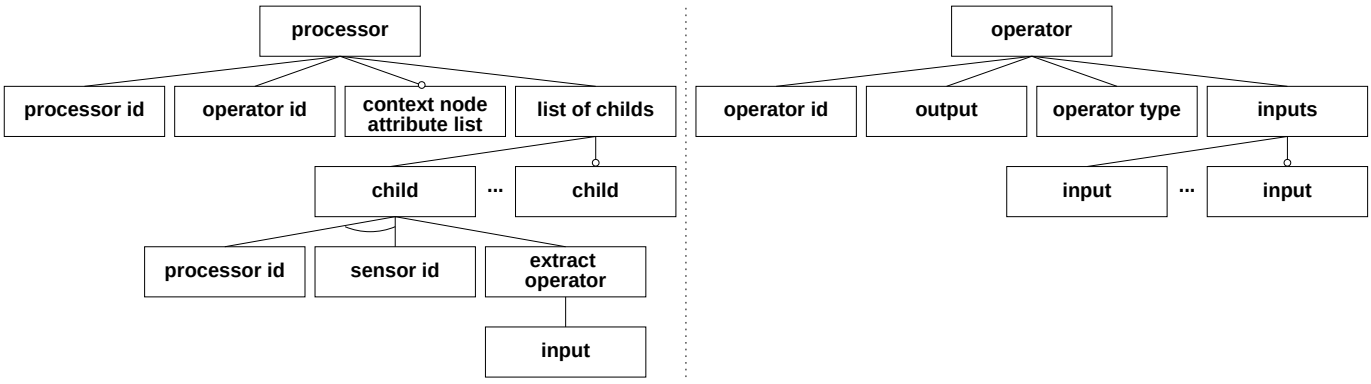
4.2 Exemple du concept politique de gestion de contexte

- Regroupement
  - Informations de contexte accédées par les clients
    $\Longrightarrow$ Nœuds de contexte visibles
  - Contrôle centralisé des ressources
    $\Longrightarrow$ Fils d'exécution et gestionnaires de rapports de contexte



4.3 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 $\Longrightarrow$ construction spécifique



5 Conception

5.1 Grammaire

5.1.1 Morceaux de messages

- Possibilité de déclarer de nouveaux types de morceaux de messages

```
chunk   ::= CHUNK chunkID'='chunkClass EOF
    CHUNK      ::= 'chunk'
    chunkID    ::= IDENT
    chunkClass ::= IDENT
    EOF        ::= ';'
```

- P.ex.,

```
chunk LinkQualityChunk = cosmos.saje.wireless.chunk.LinkQualityChunk;
```


5.1.2 Typage particulier des gestionnaires de ressources

- Possibilité de déclarer de nouveaux gestionnaires de ressources
- Cas particulier du gestionnaire de ressources dans l'architecture générale :
  - Gestionnaires de ressources ne sont pas des nœuds de contexte
  - Uniquement des morceaux de messages en sortie, aucun en entrée

```
resourceMgr   ::= RESOURCEMGR resourceMgrID '=' chunkList resourceMgrClass
                  '[' attributeRMList ']' EOF
    RESOURCEMGR      ::= 'resourcemgr'
    resourceMgrID    ::= INDENT
    chunkList        ::= '{' chunk { ',' chunk } '}'
    resourceMgrClass ::= IDENT
    attributeRMList  ::= attributeASSIGN { ',' attributeASSIGN }
    attributeASSIGN  ::= attributeID '=>' attributeValue
```

- P.ex.,

```
resourcemgr WiFiRM = {LinkQualityChunk,ESSIDChunk,AccessPointAddressChunk...}
   cosmos.saje.WiFiRM[resourceName => eth1];
```

5.1.3 Formule des capteurs

- Traduction des spécificités de l'architecture des capteurs :
  - ♦ Toujours le même opérateur $\Longrightarrow$ pas nécessaire de le nommer
  - ♦ Configuration de l'opérateur
- Distinguer la configuration du capteur de celle du gestionnaire de ressources

```
sensor   ::= SENSOR sensorMgrID '='
   (resourceMgrId | '(' resourceMgr ')') [attributeCNList] EOF
SENSOR          ::= 'sensor'
sensorMgrID     ::= IDENT
attributeCNList ::= '[' attributeCN { ',' attributeCN } ']'
attributeCN     ::= 'BO' | 'BN' | 'AO' | 'AN' | attributeASSIGN
```

- P.ex.,

```
sensor WiFi   = WiFiRM [BO, AO];
sensor Battery = ({TimeLeftChunk,ChargeLevelChunk,StatusChunk,IsChargingChunk}
                  cosmos.saje.BatteryRM) [BO, AO,AN];
```


5.1.4 Formule des opérateurs

- Possibilité de déclarer de nouveaux opérateurs
- En entrée, des messages (des ensembles d'ensembles de morceaux de messages)
- En sortie, un message

```
operator      ::= OPERATOR operatorId  '='chunkList
   operatorClass '(' chunkListList ')' EOF
OPERATOR      ::= 'operator'
operatorId    ::= IDENT
operatorClass ::= IDENT
chunkListList ::= chunkList { ',' chunkList }
```

- P.ex.,

```
operator BatteryLifeSpanCO = {BatteryLifeSpanChunk}
                             cosmos.saje.battery.operator.BatteryLifeSpanCO
                             ({StatusChunk},{TimeLeftChunk});
```

5.1.5 Formule des processeurs

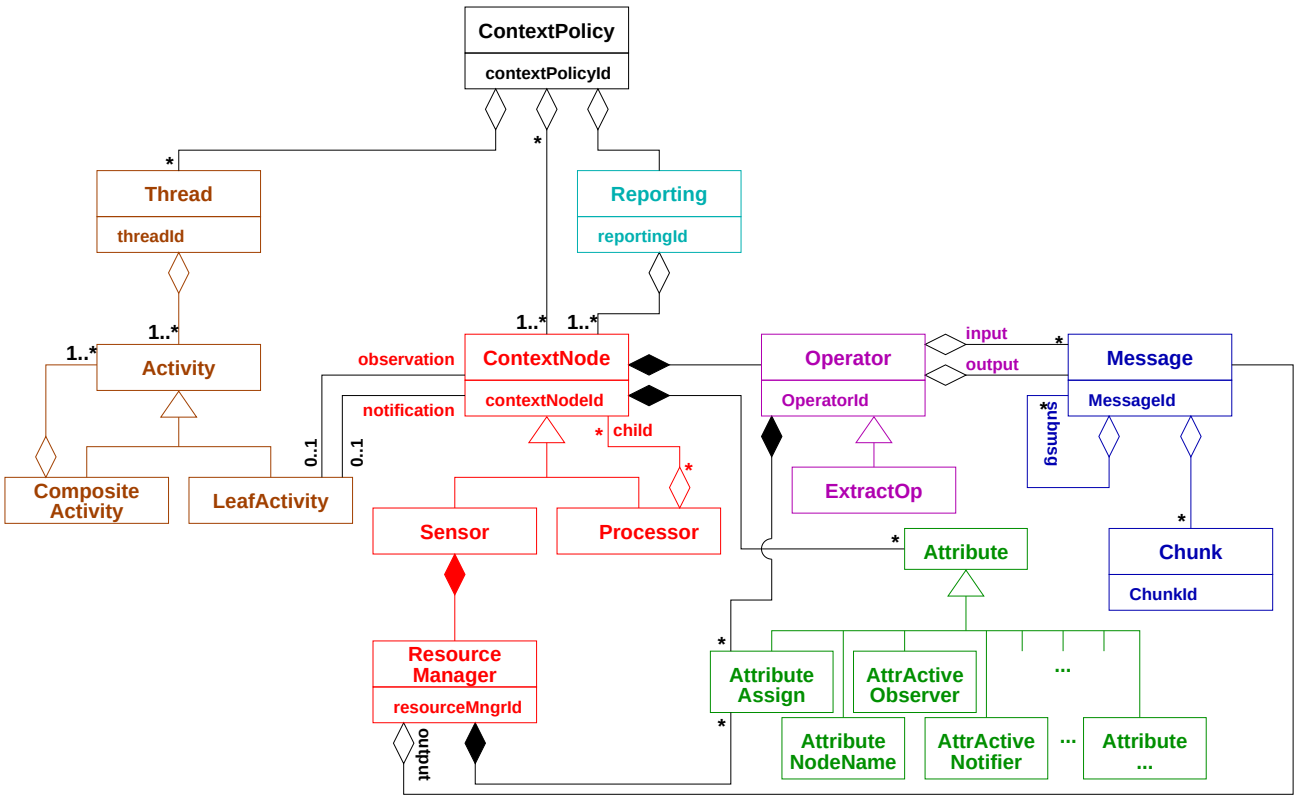
- En entrée : des messages, et en sortie : un message
- Cas particulier de l'opération d'extraction de morceaux de messages
  - ♦ Utilisation très fréquente $\Longrightarrow$ construction spécifique

```
processor   ::= PROCESSOR processorID '=' (operatorId | '(' operator ')')
                  [attributeCNList] childList EOF
PROCESSOR   ::= 'processor'
processorID ::= IDENT
childList   ::= '(' child { ',' child } ')'
child       ::= ( processorId | sensorId ) ['.' EXTRACTOP chunkList]
EXTRACTOP   ::= 'extract'
```

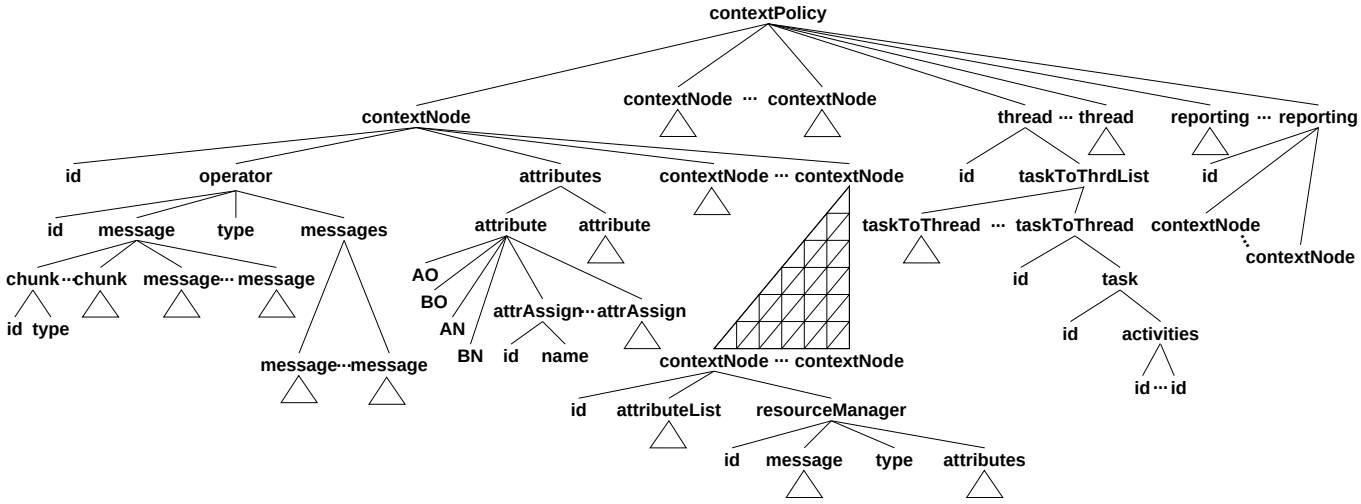
- P.ex.,

```
processor BatteryLifeSpan = BatteryLifeSpanCO(BatteryStatus,BatteryTimeLeft);
processor AverageLinkQuality = AveragerCO[nbSamples => 5]
   (WiFi.extract{LinkQualityChunk});
processor Proc = ({chunk1} identOpClass({chunk2,chunk3},{chunk4}))
   [attr=>value](child1,child2.extract{chunk4});
```


5.2 Modèle sémantique : méta-modèle de COSMOS



5.3 Arbre d'interprétation



+ Expression des idiomes qui donnent ensuite les patrons d'architecture
  - Composite, Poids-mouche, Patrons de méthodes, Singleton

5.4 Analyse sémantique

- Traduction vers Fractal ADL
- Vérifications
  - Détection d'interblocage
    - Nœuds accédés en exclusion mutuelle $\Longrightarrow$ risque d'interblocage
      - ★ Entre flots d'appels descendant (observation) et montant (notification)
    - Traduction vers un langage synchrone ?
  - Système de type de messages
    - Fractal et Fractal ADL : typage des interfaces
      - ★ Limite : composants génériques possédant tous les mêmes interfaces
    - Dream Types : typage des contenus des messages
      - ★ Vérifier qu'un morceau de message requis est offert par un enfant
    - Traduction vers Dream Types ?

6 Implantation du prototype

- Choix de JavaCC et JTB
  - COSMOS utilise Java $\Longrightarrow$ générateur du compilateur en Java
    - Choix de JavaCC + JTB plutôt que SableCC
      - ★ JavaCC : LL(k), avec JTB ou JJtree pour la génération de l'AST
        Des projets de l'écosystème FRACTAL utilisent JavaCC : p.ex., FDF [Flissi and Merle, 2006, Flissi et al., 2008]
      - ★ SableCC : LALR(1), donc plus puissant, mais moins performant
- Choix de JTB plutôt que JJTree
  - Génération des classes du compilateur dans un paquetage séparé
  - Non-nécessité de modifier les classes du compilateur
    - Utilisation du patron de conception Visiteur
  - Patron Visiteur généré et extension simple à réaliser
  - Méthodes `visit()` des classes du patron générées
    - Commentaires avec la production de la grammaire concernée

7 Évaluation, discussion

- Facilité d'utilisation
  - Langage conçu suite à une analyse de domaine
  - Idiomes correspondant aux patrons architecturaux dans l'architecture des politiques de contexte de COSMOS
  - Modularité des expressions
- Extensibilité : déclaration modulaire
  - À venir, espace de nommage
- Analysabilité : typage des éléments, idiomes
- Réutilisation systématique : idiomes
- Sûreté : typage des éléments
- Composabilité : opérateur, entrée/sortie
- Éléments à évaluer avec une implantation complète
  - Testabilité
  - Optimisation

8 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, répertoire `sandbox/leon/cosmosdsl`
    - Projet Maven, tests unitaires avec JUnit
  - Langage spécifié en JavaCC, tests unitaires effectués
- Travaux en cours et futurs
  - Formules des extra-fonctionnalités : gestions des activités et des messages
  - Transformation de l'arbre de syntaxe en arbre d'interprétation
  - Analyse sémantique :
    - Génération d'architecture FRACTAL ADL
    - Détection d'inter-blocage
    - Système de types de messages

Références

[Bontemps et al., 2004] Bontemps, Y., Heymans, P., Schobbens, P.-Y., and Trigaux, J.-C. (2004). Semantics of Feature Diagrams. In Männistö, T. and Bosch, J., editors, *Proc. of Workshop on Software Variability Management for Product Derivation (Towards Tool Support)*, Boston, USA.

[Botts and Robin, 2007] Botts, M. and Robin, A. (17-07-2007). Sensor Model Language (SensorML). Technical report, University of Alabama in Huntsville.

[Boutros Saab et al., 2002] Boutros Saab, C., Bonnaire, X., and Folliot, B. (2002). PHOENIX : A Self Adaptable Monitoring Platform for Cluster Management. *Cluster Computing*, 5(1) :75–85.

[David and Ledoux, 2005] David, P.-C. and Ledoux, T. (2005). WildCAT : a generic framework for context-aware applications. In *Proc. 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing*, pages 1–7, Grenoble (France).

[Flissi et al., 2008] Flissi, A., Dubus, J., Dolet, N., and Merle, P. (2008). Deploying on the Grid with DeployWare. In *Proc. 8th IEEE International Symposium on Cluster Computing and the Grid*, Lyon (France).

[Flissi and Merle, 2006] Flissi, A. and Merle, P. (2006). A Generic Deployment Framework for Grid Computing and Distributed Applications. In *Proc. 2nd International OTM Symposium on Grid computing, high-performAnce and Distributed Applications*, Lecture Notes in Computer Science, pages 1402–1411, Montpellier (France). Springer-Verlag.

[Frølund and Koistinen, 1998] Frølund, S. and Koistinen, J. (1998). QML : A Language for Quality of Service Specifiation. Technical report.

[Haghighi et al., 2006] Haghighi, P., Zaslavsky, A., and Krishnaswamy, S. (2006). An Evaluation of Query Languages for Context-Aware Computing. In *Proc. of the 17th IEEE International Conference on Database and Expert Systems Applications*, pages 455–462, Washington, DC, USA.

[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).

[Klyne et al., 2007] Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M., , and Tran, L. (2007). Composite Capability/Preference Profile (CC/PP) : Structure and vocabularies 2.0. Technical report, Technical report, W3C recommandation.

[Mernik and Sloane, 2005] Mernik, M. H. J. and Sloane, A. (2005). When and How to Develop Domain-Specific Languages. *ACM Computing Surveys*, 37(4) :316–344.

[Open Mobile Alliance, 2001] Open Mobile Alliance (2001). User Agent Profile. WAP Forum.

[Parsons, 2001] Parsons, S. (2001). Wurfl : Wireless Universal Resource File. http:wurl.sourceforge.net.

[Román et al., 2002] Román, M., Hess, H., Cerqueira, R., Ranganathan, A., Campbell, R., and Nahrstedt, K. (2002). Gaia : A Middleware Platform for Active Spaces. *SIGMOBILE Mobile Computing Communication Review*, 6(4) :65–67.