Projet CMCU, Action Intégrée n° 05G 1409

Architectures dynamiques dans le contexte des applications à base de composants et orientées services

Département d'Informatique et de Mathématiques Appliquées

Laboratoire d'Analyse et d'Architecture des Systèmes

Mohammed Karim Guennoun

Directeurs de thèse: Khalil Drira, Michel Diaz

Contexte

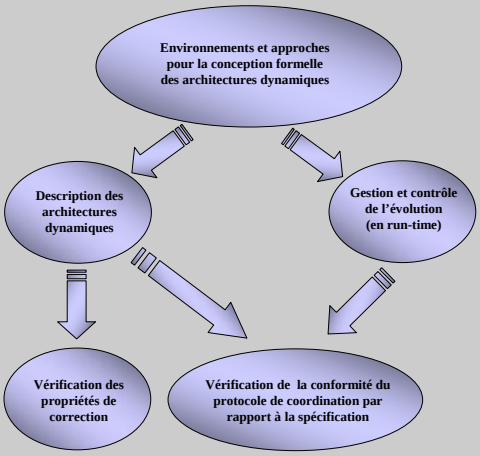
- Adaptabilité des architectures :
  - Spécification.
  - Contrôle.
  - Gestion.
- Architectures logicielles dynamiques distribuées orientées composants.
- Styles d'architectures.

Des outils pour la description et la gestion des architectures dynamiques

- Coordination dans les applications de grande taille
- Consistance et propriétés de correction
- Favoriser l'adaptabilité des applications et services

Objectifs

- Spécification formelle du style de l'architecture du système:
  - Décrire les instances consistantes de l'architecture.
- Contrôle de l'évolution dynamique en ligne:
  - Autoriser ou non une évolution de l'architecture.
- Gestion (consistante) de l'évolution dynamique en ligne:
  - Spécifier les opérations à effectuer en réponse à un besoin d'adaptabilité de l'architecture (oui/non + comment?).



Approche

Choix du formalisme

- Notre choix : les grammaires de graphe:
  - Une architecture = un graphe. un composant = nœud, une connexion ou une interdépendance = un arc).
  - Moyen formel et opérationnel pour la vérification et la validation.
- Les autres possibilités: ADLs (e.g. Wright +CSP, Darwin +pi-calculus):
  - Langages formels pas appropriés pour la phase de spécification (différents profils des interlocuteurs).
  - Traitement limité de l'évolution dynamique de l'architecture.

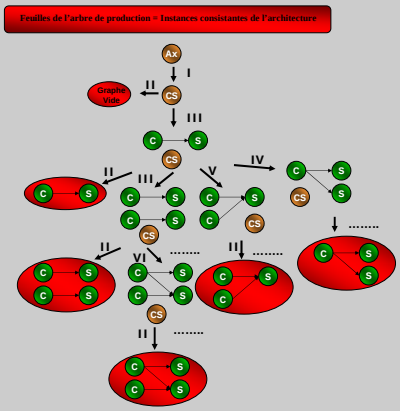
Formalisation

- Une instance de l'architecture = un graphe.
- Un style d'architecture = une grammaire de graphe.
- Une propriété architecturale = existence ou absence d'un homomorphisme de graphes
- Une architecture dynamique = un graphe initial + un ensemble de règles de transformation.
- Modèle pour l'analyse de l'architecture (passage d'un point de vue ou d'un niveau d'abstraction à un autre) = grammaire de graphe NCE.

Description du style d'architecture (le style 2-tier)

Grammaire de graphe < AX;{CS};{Client,Serveur}; Prod}, où Prod =

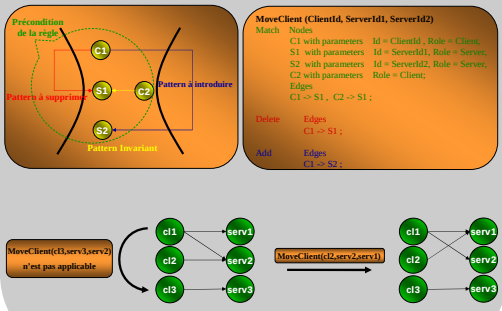
- < {AX};{ };{CS};{ } > (I)
- < {CS};{ };{ };{ } > (II)
- < { }; {CS};{Client(c), Serveur(s), c -> s} > (III)
- < { }; {CS, Client(c)}; {Serveur (s), c-> s} > (IV)
- < { }; {CS, Server(s)}; {Client(c), c->s} > (V)
- < { }; {CS, Client(c), Server(s)}; {c->s} > (VI)



Evolution de l'architecture (les règles de transformation)

- Une règle de transformation est décrite par un graphe partitionné.
- La partition du graphe de la règle spécifie :
  - Quand une évolution est possible :
    - Exigeant la présence d'un Pattern dans le graphe représentant l'architecture courante de l'application.
  - Et comment l'évolution est réalisée :
    - Transformation du graphe de l'architecture par l'ajout et la suppression de Patterns.
    - Génération des actions basiques pour la modification de l'architecture.
- L'application de la règle est subordonnée à l'existence d'homomorphisme entre sa précondition et le graphe modélisant l'architecture courante.

Exemple: Rediriger un client d'un serveur vers un autre dans une application à architecture Client/Serveur



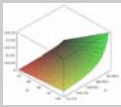
Applications et cas d'étude traités

- Caractérisation de styles architecturaux de base, introduction des règles de coordination correspondantes, et vérification formelle de leur consistance:
  - Style 2-tier (Client/serveur classique).
  - Style 3-tier (Client/ Middleware/ Serveur).
  - Styles pour les architectures orientées services (avec les notions d'orchestration de chorégraphie).
- Modélisation d'applications coopératives distribuées, Introduction des règles de coordination pour gérer l'adaptation par rapport aux contraintes du contexte d'exécution tout en respectant les exigences du contexte d'utilisation :
  - Edition Coopérative
  - Revue Coopérative
  - Opérations d'intervention d'urgence

Réalisations

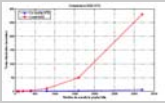
Algorithmes de recherche d'homomorphismes

- Première implémentation en JAVA, avec un algorithme récursif :
  - ☺ *Variables dans les labels.*
  - ☺ *Labels de type string pour les nœuds de la règle.*
  - ☹ *Limitation par rapport à la structure de la règle (pas de cycles dans la précondition).*
  - ☹ *Passage à l'échelle laborieux.*
- Deuxième implémentation en C++, avec un algorithme par décomposition :
  - ☺ *Aucune limitation au niveau de la structure des graphes.*
  - ☺ *Liste de labels typés.*
  - ☺ *Traitement d'arcs valués (permet la distinction du type d'interdépendance).*
  - ☺ *Complexité au pire cas de:* $\left\{ K^2 \sqrt{\frac{N_G}{N_G - N_{PRE}}} \times \left(\frac{N_{PRE}}{N_G - N_{PRE}} \right) N_G^{N_{PRE}+2} \right\}$
- Troisième implémentation avec un algorithme profitant du contexte d'utilisation du formalisme (ensemble stable de règles potentiallment applicables:
  - ☺ *Mise à jour de l'arbre de recherche au lieu de sa reconstruction à chaque transformation*
  - ☺ *Gain important par rapport à l'algorithme précédent*



Implémentation

- Deux modules en couches:
  - Moteur de transformation de graphes
  - Protocole de reconfiguration programmable
- API C++ || API Java
- Résultats expérimentaux pour une règle de 10 nœuds
  - Graphe de 1 000 nœuds : 0.5 s
  - Graphe de 1 000 000 nœuds : 2 min
- Comparaison expérimentale avec l'outil AGG (Technische Universität Berlin):



Conclusion

- Modèle basé sur les grammaires de graphes pour :
  - La description des architectures dynamiques.
  - Le contrôle et la gestion de ces architectures.
  - Vérification et validation des propriétés architecturales.
- Algorithmes de recherche d'homomorphismes et de transformation de graphes appropriés pour le type de problématique à traiter.
- Outil basé sur la réécriture de graphes pour la conception/validation et pour le contrôle/gestion des architectures logicielles dynamiques.
- Application à des styles architecturaux génériques et à différentes applications coopératives

Perspectives

- Approches et méthodes:
  - Coordination des modèles et des procédures d'adaptabilité architecturale et comportementale
  - Coordination et optimisation de l'adaptation multi-niveaux
- Applications:
  - Coordination de la reconfiguration architecturale dans les applications à base de services Web avec l'adaptation comportementale au niveau des processus BPEL.
  - Coordination de la reconfiguration niveau application avec des mécanismes d'adaptation niveau transport.
- Algorithmes et outils:
  - Implantation du deuxième algorithme
  - Intégration générique avec les technologies de programmation distribuée (Web services, Composants)

Publications

- 13 articles (actes et comité de lecture):
  - 3 revues internationales:
    - ✓ K. Guennoun, K. Drira and M. Diaz. Considering topological constraints for the description of dynamic service-oriented orchestrated architectures. *International Journal of Computing & Information Sciences (IJCIS)*, Vol.5, N°1, pp.45-50, Avril 2007
    - ✓ C. Chassot, K. Drira, K. Guennoun, F. Armando, E. Exposito and A. Lozes. Towards autonomous management of QoS through model-driven adaptability in communication-centric systems. *International Transactions on Systems Science and Applications*, Vol.2, N°3, pp.255-264, 2006.
    - ✓ K. Guennoun and K. Drira. Using graph grammars for interaction style description. Application to service-oriented architectures. *International Journal of Computer Systems Science & Engineering*, Vol.21, N°4, pp.293-299, Juillet 2006
  - 1 revue Francophone
  - 7 Conférences internationales
  - 2 Conférences francophones