



# Laboratoire LIUPPA – Equipe Self-\*

Université de Pau et des Pays de l'Adour

Eric Cariou



# Laboratoire LIUPPA

- Laboratoire d'Informatique de l'Université de Pau et des Pays de l'Adour (LIUPPA)
  - Equipe d'accueil (EA 3000)
  - 55 membres  
(32 enseignants/chercheurs, 23 doctorants)
- Structuration en 2 axes et 5 équipes-projets
  - Génie logiciel et systèmes distribués
  - Traitement de l'information et de l'interaction
- Recherche appliquée dans domaines variés
  - Génie logiciel, agents et composants logiciels, documents électroniques, visualisation scientifique, sécurité informatique ...

# Ingénierie des modèles au LIUPPA

- Thématique transversale à plusieurs équipes
  - En tant qu'outil et base de conception pour
    - Environnement Informatique d'Apprentissage Humain (EIAH)
    - Ingénierie de systèmes complexes distribués
  - Travaux sur l'IDM en génie logiciel
    - Equipe-projet Self-\* du LIUPPA
    - 7 permanents (3 PR, 4 MC dont 1 HDR)
      - Philippe Aniorté, Franck Barbier, Nicolas Belloir, Jean-Michel Bruel, Eric Cariou, Nabil Hameurlain, Congduc Pham
    - 5 thésards
      - Xavier Aretxandieta, Cyril Ballagny, Xavier Elkorobarrutia, Fabien Roméo, Séverine Sentilles

# Equipe-projet Self-\* du LIUPPA

- Thématique générale
  - A partir d'une approche orientée modélisation
    - Convergence agent/composant/service
    - Entités logicielles autonomes et dynamiquement composables (services autonomiques)
  - Domaines d'application de prédilection : systèmes embarqués, mobiles, sans fil...
- 3 aspects « IDM »
  - Exécutabilité de diagrammes d'états UML 2.0
  - Composition de comportement
  - Gestion de liens de composition

# Exécutabilité de modèles : PauWare

- Moteur exécution de machines à états UML
  - Comportement d'un composant spécifié par des machines à états et diagrammes de séquence
- Principes
  - Moteur/librairie PauWare en Java : gestion de machines à états à l'exécution
    - Gestion de l'exécution du composant via son modèle de comportement intégré
  - «Isomorphisme» modèle UML et code PauWare
    - Générateur XMI 2.0 vers Java/PauWare
  - Moteur implémente sémantique UML 2.0

# Exécutabilité de modèles : PauWare

- Exécution de modèle
  - Le code Java/PauWare exécuté est le modèle
- Simulation/vérification de modèle
  - Via l'exécution du code PauWare
  - Administration/envoi d'événements de tests possible via JMX et interface distante Web
    - Offre un support pour le test de composants
- Deux mises en œuvre
  - PauWare.Composytor : pour J2SE et J2EE
    - Intégration de machines à états dans composants EJB
  - PauWare.Velcro : pour J2ME
    - Pour appareils types PDA, téléphones mobiles ...
    - Avec création de WMX (Wireless Management eXtensions) : extension de JMX pour environnement mobile et sans fil

# Exécutabilité de modèles : PauWare

- Self-healing (auto-réparation)
  - Si transition d'états déclenche une erreur, possibilité de revenir à l'état précédent
    - Sauf si violation d'invariants
- Self-configuration (auto-configuration)
  - Forcer le passage à un état (aussi «complexe» soit-il) donné
- Conclusion sur PauWare
  - Toutes ces fonctionnalités/caractéristiques possibles car support/intégration/réification du modèle dans le code

# Composition de comportement

- Principe général
  - Extension des fonctionnalités de composants COTS par approche orientée aspect
    - Une fonctionnalité = un aspect
    - Ex: interface d'administration/test de PauWare
  - Comportement décrit par des machines à états
    - Comportement du composant de base
    - Comportement de chaque fonctionnalité/aspect
  - Tissage des aspects sur le composant
    - Comportement global obtenu = machine à état résultant de la composition des machines à états du composant et des aspects
- Travaux en cours
  - Définition langage de composition de machines à états et de règles de composition

# Gestion de liens de composition

- Base théorique
  - UML 1.X & 2.0
    - Agrégation/composition & ports/structures composites/délégation → relativement mal définis
  - Spécification de relations « tout – partie »
    - 4 types de composition hiérarchique
    - Aspects statiques et dynamiques (encapsulation, partageabilité, cycle de vie, ...)
- Réalisation
  - Mise en œuvre de ces concepts de composition de la spécification à l'implémentation

# Gestion de liens de composition

- Profil UML WPCP
  - Whole-Part Composition Profile
  - Spécification des 4 relations de composition via profil UML + contraintes OCL
- Ensemble de transformations de modèles
  - Pour génération finale de code Fractal
  - Ajout d'un nouveau contrôleur Fractal
    - Gestion des liens de composition à l'exécution
  - Mise en œuvre dans Objecteering / UML 1.5
- Perspectives
  - Validation des contraintes OCL spécifiant les liens de composition sur les modèles transformés
  - Génération de code pour autres modèles de composant

# Pour plus d'informations ...

---

- PauWare
  - <http://www.pauware.fr>
- WMX
  - <http://www.univ-pau.fr/~fromeo/wmx/>