

# Architectures à composants et agents pour la conception d'applications réparties adaptables

Sébastien Leriche

Equipe LYRE – ingénierie des Langages  
pour les sYstèmes Répartis et Embarqués

Directeur de thèse : Jean-Paul Arcangeli



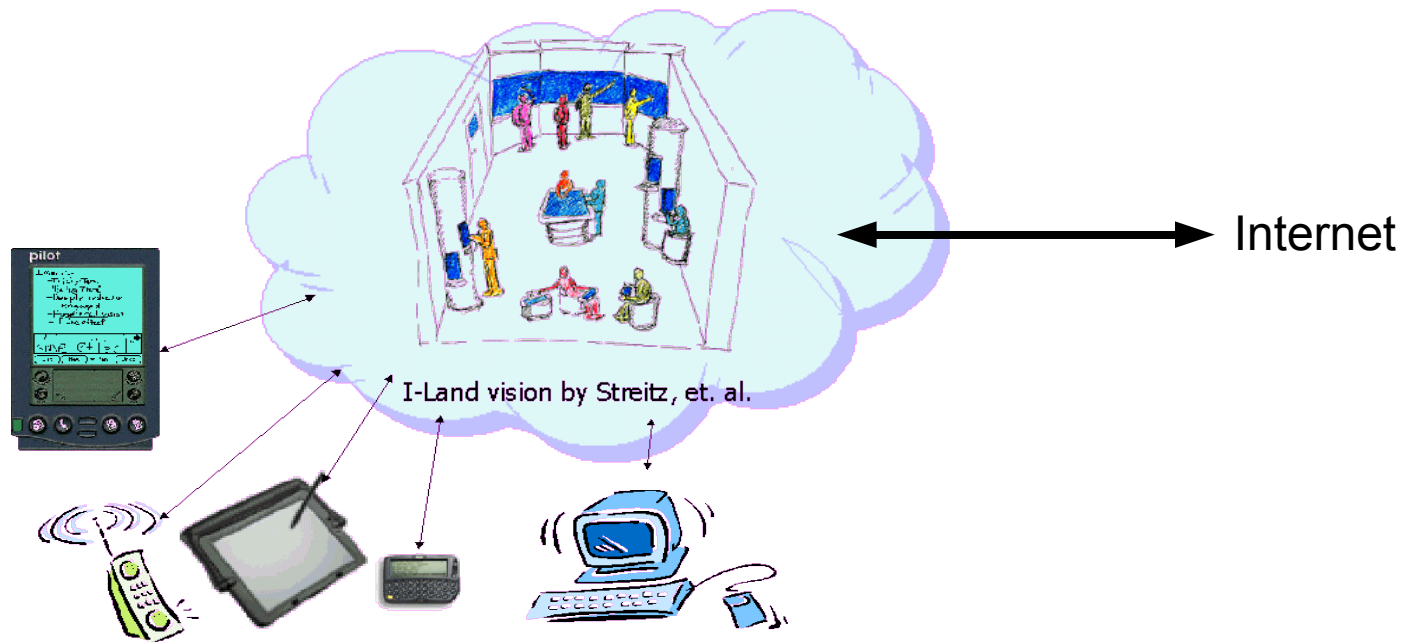
# Plan

- I. Problématique et objectifs
- II. Modèle d'agent mobile adaptable  
micro-architecture
- III. Patron de conception des systèmes P2P « purs »  
macro-architecture
- IV. Modèle d'agent flexible  
micro-architecture
- V. Conclusion & perspectives

# I. Problématique et objectifs

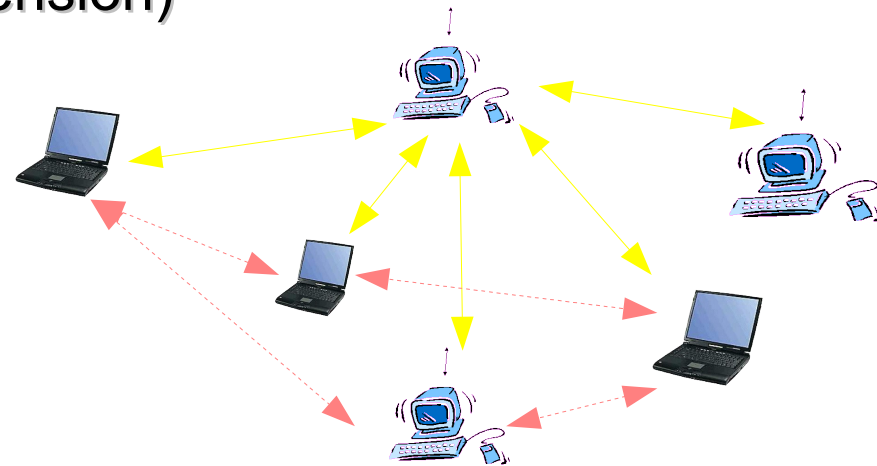
# Contexte

- Informatique répartie
  - Interconnexion de systèmes variés (portable, PDA, PC...)
  - Répartition à grande échelle
  - Entre la grille et les systèmes pervasifs / ubiquitaires



# Problématique (1)

- Exemple : Mutualisation de ressources
- Caractéristiques
  - Hétérogénéité, variation des conditions d'exécution
  - Administration indépendante (sites distincts)
  - Contrôle décentralisé
  - Facteur d'échelle (unités, dimension)
  - Ouverture et évolutivité
- Autres exemples
  - Calcul réparti
  - Distribution de logiciel
  - Services embarqués



# Problématique (2)

- Besoin d'autonomie
  - Support des variations de l'environnement
- Besoins de flexibilité
  - Configuration, extensibilité et adaptation dynamique
  - Personnalisation
- Implication sur la conception
  - Recherche de ressources
  - Organisation des traitements répartis
    - Déploiement automatique
  - Sûreté et sécurité

# Objectifs

- Réduction de la complexité
  - Prise en compte des difficultés identifiées
  - Technologies logicielles pour le développement
    - Intergiciel (*middleware*)
    - Modèle de programmation
    - Environnement de développement
  
- Les technologies existantes ne répondent pas correctement aux besoins
  - Proposition d'autres solutions

# Technologies - P2P « pur »

## ■ Justification

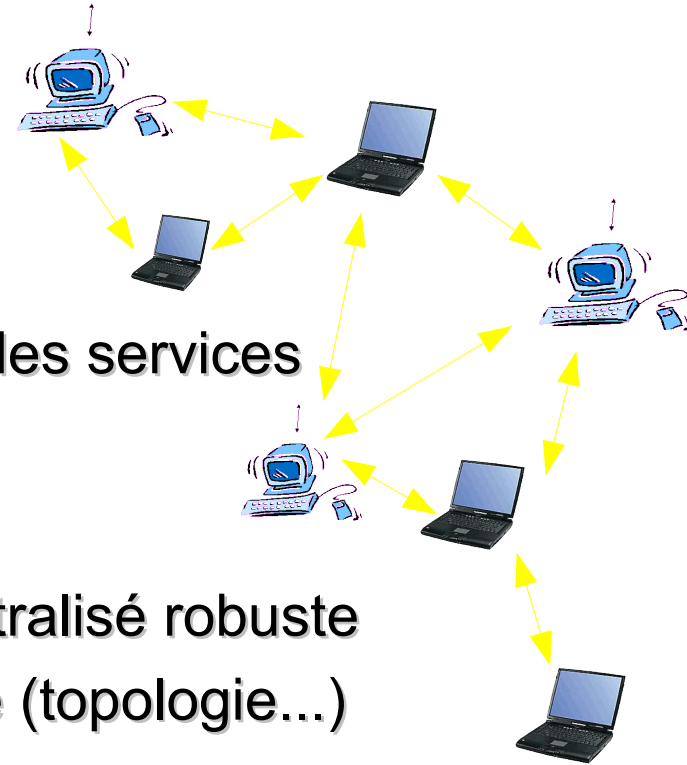
- Approche centralisée irréaliste
- Pas de serveurs lourds
- Chaque entité peut offrir et exploiter des services
- Modèle P2P « pur »

## ■ Apports

- Modèle d'organisation répartie décentralisé robuste
- Adapté aux évolutions et à l'instabilité (topologie...)

## ■ Limites

- Construction d'applications sur ce modèle difficile
- Besoin d'outils spécifiques pour le développement

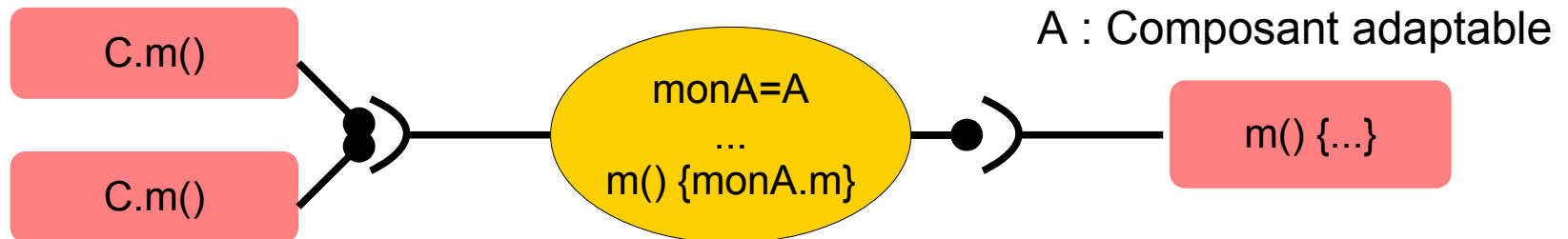
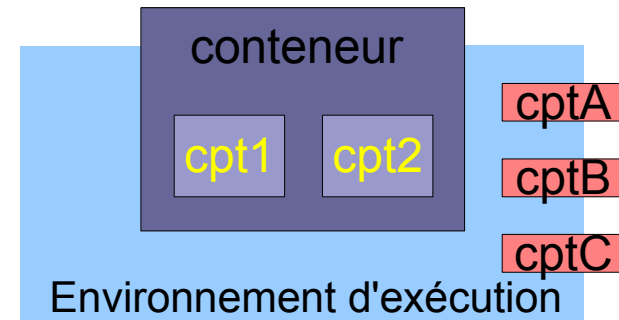


# Technologies – Composants (1)

## ■ Apports

- Structuration, réutilisation, composition
- Séparation des préoccupations
  - Aspects métiers vs. non-fonctionnels
  - Adaptation statique
- Simplification du développement et de la maintenance
- Approche composant-connecteur (adaptation dynamique)

Approche composant-conteneur



C : Connecteur

# Technologies – Composants (2)

## ■ Limites

- Implantés dans des serveurs d'application
  - Lourds, peu adaptés à des petits systèmes
  - Déploiement non trivial et non automatique
- Modèles d'assemblages préconçus
  - Environnement d'exécution spécialisé (CCM, EJB, Fractal...)

→ Concept de composant

# Technologies – Agents (1)

- Notre définition (agent logiciel) :
  - Entité autonome capable de communiquer
  - Dispose de connaissances et d'un comportement privés
  - Capacité d'exécution et proactivité
  
- Agent = objet + autonomie

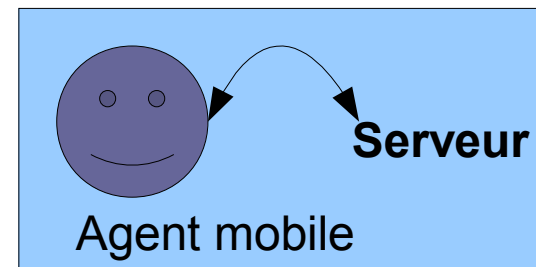
# Technologies – Agents (2)

- Agent mobile : agent logiciel + mobilité
  - Code + données + état d'exécution
  - Déplacement des traitements
  - Complète les capacités de communication
    - La mobilité est une propriété
  - Outil d'adaptation (réactivité, proactivité, suivi d'évolutions...)

Déconnexion possible



Communications locales,  
filtrage, adaptation du flux



# Technologies – Agents (3)

## ■ Apports

- Principalement sur le plan génie logiciel
- Unité de structuration des applications
- Unification de différents paradigmes d'interaction :
  - Client/Serveur – communication par message – code mobile
- Paradigme adapté au contexte
  - Outil de conception
  - Approche de haut niveau de la répartition
  - Autonomie et proactivité pour la réactivité

## ■ Limites

- Sécurité, nouveau paradigme...

# **II. Modèle d'agent mobile adaptable Micro-Architecture**

# Justification

- Utilisation d'agents mobiles comme support d'exécution des applications réparties
  - Utilisables dans le contexte de grande échelle proposé
  - Adaptation au contexte d'exécution variable
  
- Besoin d'adaptation dynamique
  - Lors de la mobilité
    - Adaptation des protocoles (localisation, communication...)
    - Adaptation des services non-fonctionnels
  - Lors de changements dans l'environnement
    - Services, périphériques, mobilité physique...

# Principes architecturaux (1)

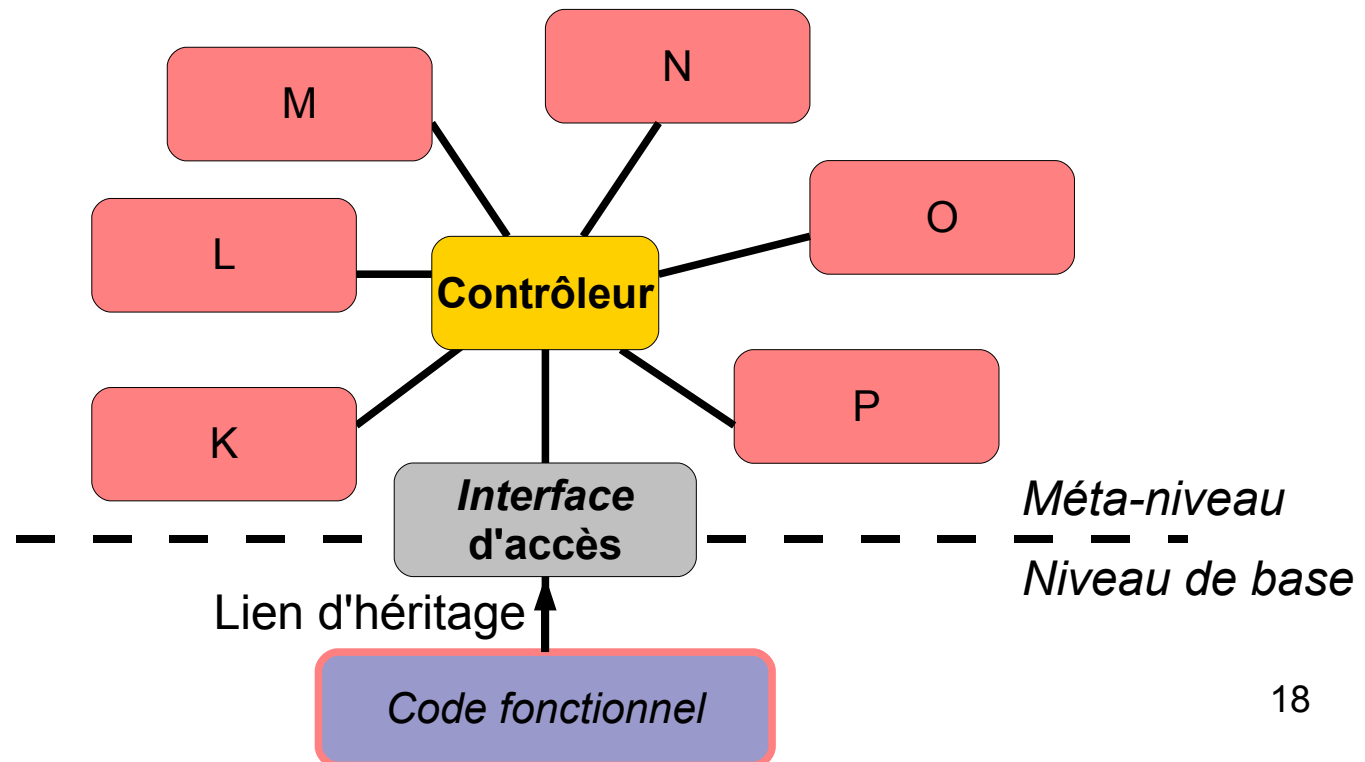
- Séparation des préoccupations et modularité
- Technologies à base de composants
  - Réutilisation, mécanismes d'adaptation dynamique
  - Grain fin pour les éléments adaptables (micro-chirurgie)
- Propriétés
  - Transparence
  - Adaptation individuelle

# Principes architecturaux (2)

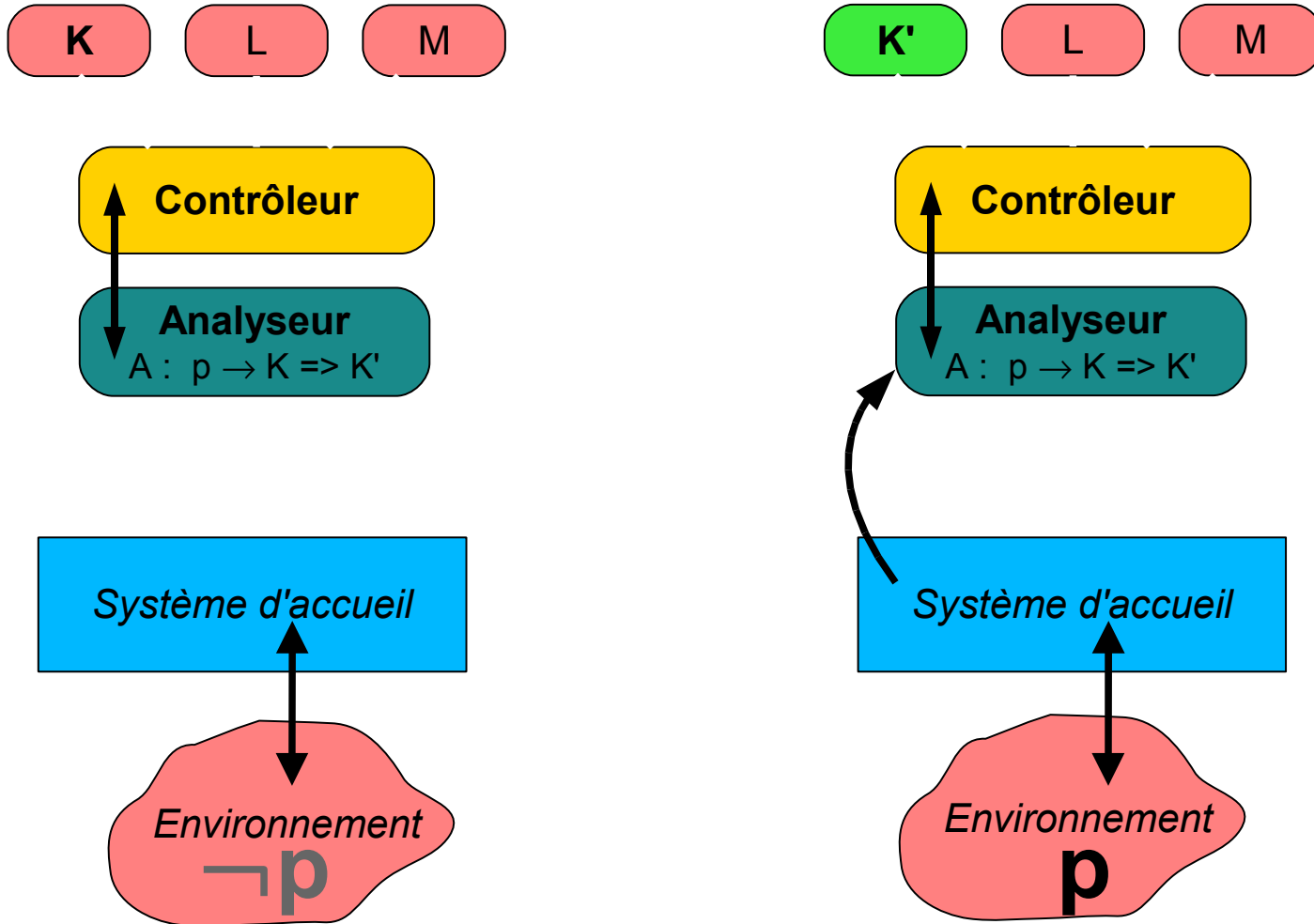
- Architecture à méta-objets
  - Code fonctionnel (comportement) au niveau de base
  - Mécanismes non-fonctionnels au niveau méta
  - Manipulation des mécanismes dans le méta-niveau
  - Inspirée de CodA [McA95] – PlasmaR [Mig98]
  
- Micro-composants remplaçables dynamiquement ( $\mu$ C)
  - Un micro-composant par mécanisme non-fonctionnel
    - Cycle de vie, communication...
  - Un agent est alors composé :
    - D'un comportement fonctionnel au niveau de base
    - D'un assemblage de micro-composants (méta-niveau)

# Principes architecturaux (3)

- Ajout d'un connecteur (*Contrôleur*)
  - Pour permettre l'adaptation dynamique des  $\mu$ C
  - Structure en étoile (connecteur unique)

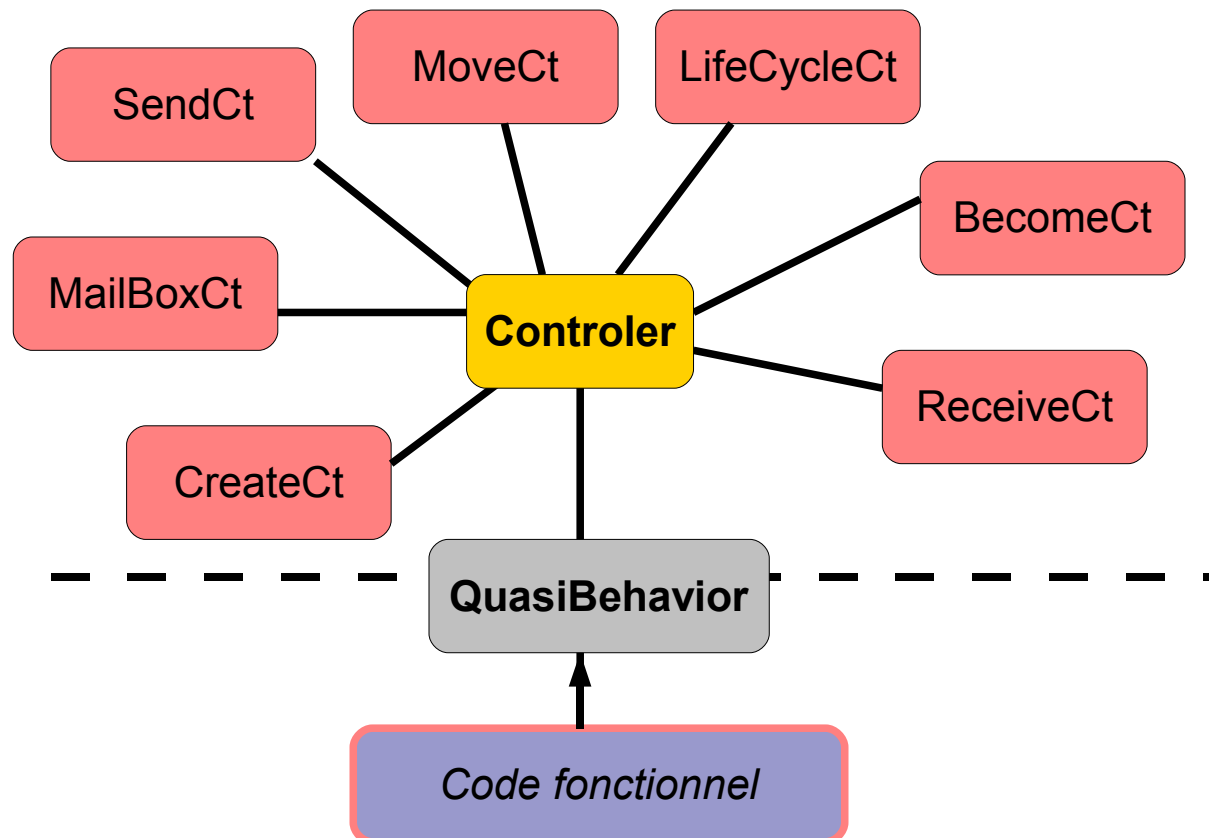


# Adaptation dynamique (1)



# Architecture d'agent mobile adaptable

- Basée sur le modèle d'acteur [Hew77, Agh86]
  - Modèle concret, sémantique claire



# Du modèle à JavAct v0.5

- Prototype JavAct<sup>δ</sup>
  - Modèle entièrement développé et testé
  - Simulation des sondes du système d'accueil
  - Analyseurs simples
  
- JavAct v0.5 (intégration aux travaux d'équipe)
  - Reprise du prototype sans les parties simulées
  - Licence LGPL
  - « Testé » et utilisé en enseignement
    - TP et projets de niveau Master
  - Plugin Eclipse pour simplifier le développement
  - <http://javact.org>

# Conclusion

- Architecture d'agent permettant l'adaptation dynamique
  - Adaptation fine, conservation de l'autonomie
- Implémentation complète
  
- Travaux connexes
  - ProActive, Aglets, Jade, MadKit...
  - Pour certains, adaptation des services fonctionnels
  - Pas ou peu adaptés aux contextes fortement répartis

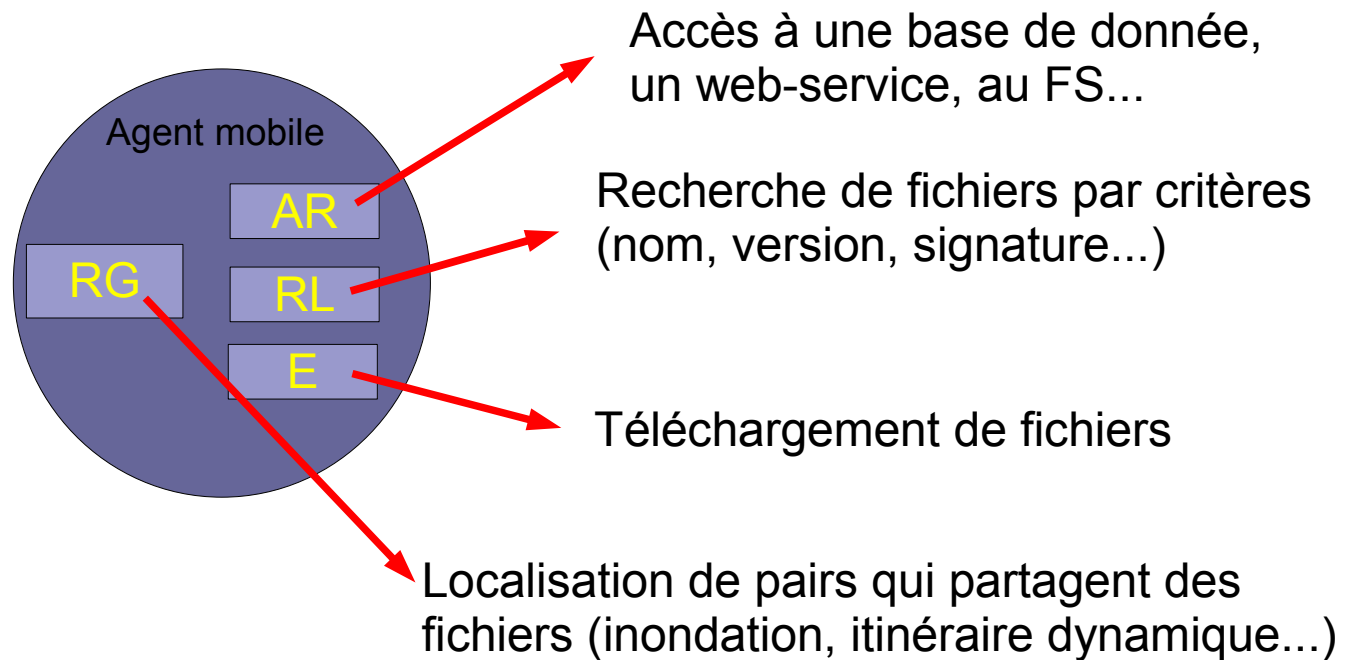
# **III. Patron de conception pour les systèmes P2P « purs » Macro-Architecture**

# Éléments constitutifs d'un système P2P

- Éléments génériques
  - Recherche globale
  - Recherche locale
  - Méta-informations
  - Exploitation des ressources
  - Accès aux ressources
  
- Implémentation possible sous la forme de composants
  - Réutilisation, composition
  - Comment les déployer ?
  - Quel environnement d'exécution ?

# 1<sup>ère</sup> approche

## ■ Scénario de mutualisation de fichiers



## ■ Réutilisation possible de certains composants

# Patron de conception

Pair client



Pair serveur



Ressources / Services

Méta-informations  
- pairs connus  
- ressources locales

Méta-informations  
- pairs connus  
- ressources locales

Agent d'Accès  
aux Ressources

Agent de  
Recherche  
Locale

Agent d'  
Exploitation

Agent mobile

Agent mobile

RG

AR

RL

E

RG

AR

RL

E

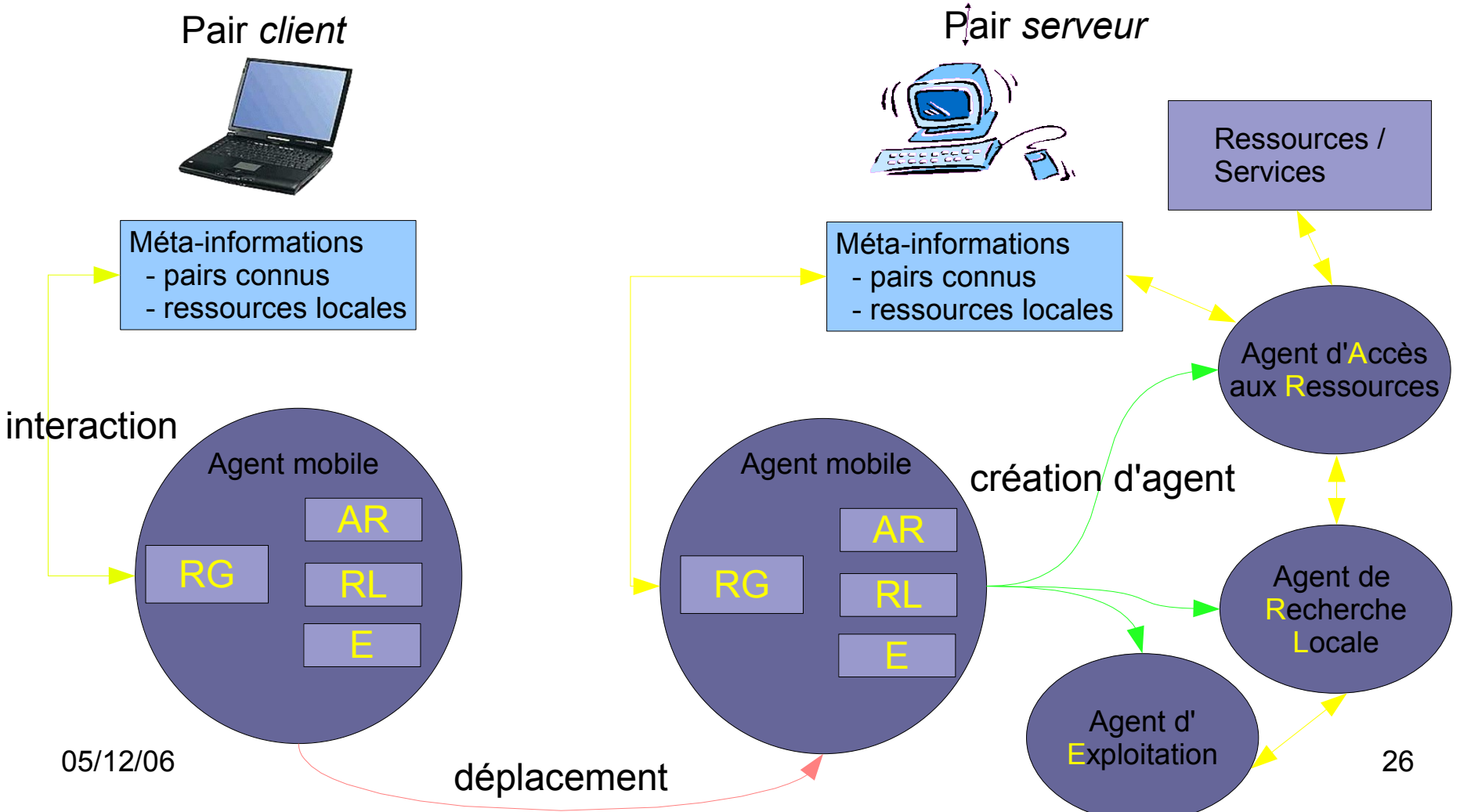
création d'agent

déplacement

interaction

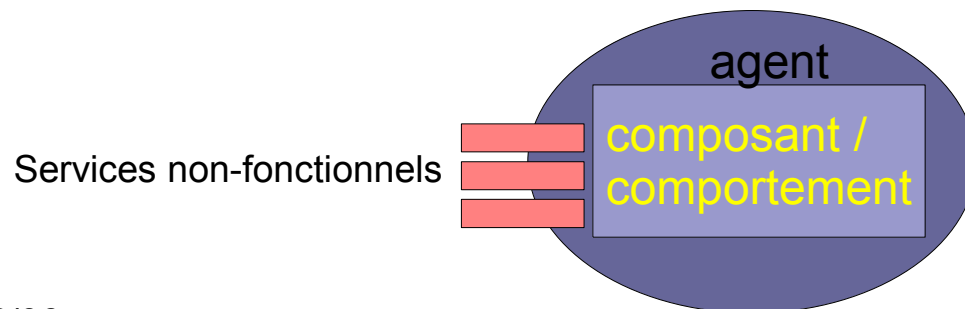
05/12/06

26



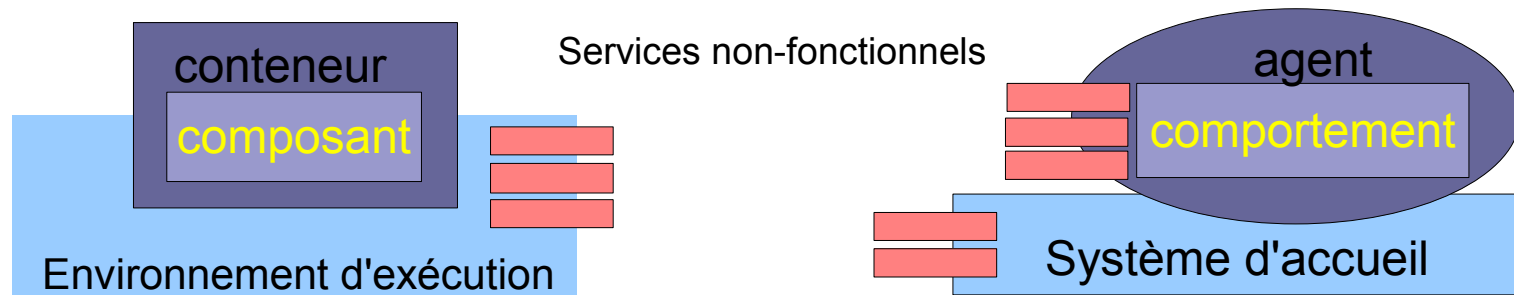
# Déploiement des composants

- Utilisation d'agents mobiles
  - Localisation robuste
  - Personnalisation et déploiement personnalisé
  - Composant P2P → Comportement d'agent
  - Avantages en terme de modèle de programmation
  - Utilisation des services fournis par les micro-composants
    - En particulier de la mobilité
    - Uniformité et puissance du modèle



# Adaptation pour les composants

- Parallèle avec le modèle composant-conteneur
  - Agent = conteneur actif mobile



- Utilisation d'agents mobiles adaptables
  - Adaptation d'un composant via l'adaptation de l'agent
  - Propriétés d'adaptation similaires (transparence, autonomie...)

# Conclusion

- Patron de conception
  - Composants logiciels
  - Déploiement et adaptation par des agents mobiles adaptables
  - Modèle P2P
- Implémentation « JavAne »
  - Framework Java
  - Prototypes logiciels complets réalisés par des étudiants
  - Réutilisation de composants « P2P » et des  $\mu$ C de JavAct
- Travaux connexes
  - M3, ACP2P, Software Dock
  - Pas d'association agent-composant-P2P

# **IV. Modèle d'agent flexible**

# Besoins d'adaptation supplémentaire (1)

- Modèle d'agent mobile adaptable : architecture figée
  - Choix des micro-composants lié au modèle d'acteur
  - Pas de possibilité d'ajout ou de suppression de  $\mu\text{C}$
- Historique du développement de JavAct
  - Ajout d'un nouveau  $\mu\text{C}$  lors de l'ajout de la mobilité
  - Modification des interfaces pour la comm. avec retour
  - Architecture mal adaptée pour l'implémentation d'AMAS
- L'architecture ne permet pas d'intégrer de nouveaux  $\mu\text{C}$  et n'est pas minimale

→ Architecture plus flexible

# Peut-on aller plus loin ?

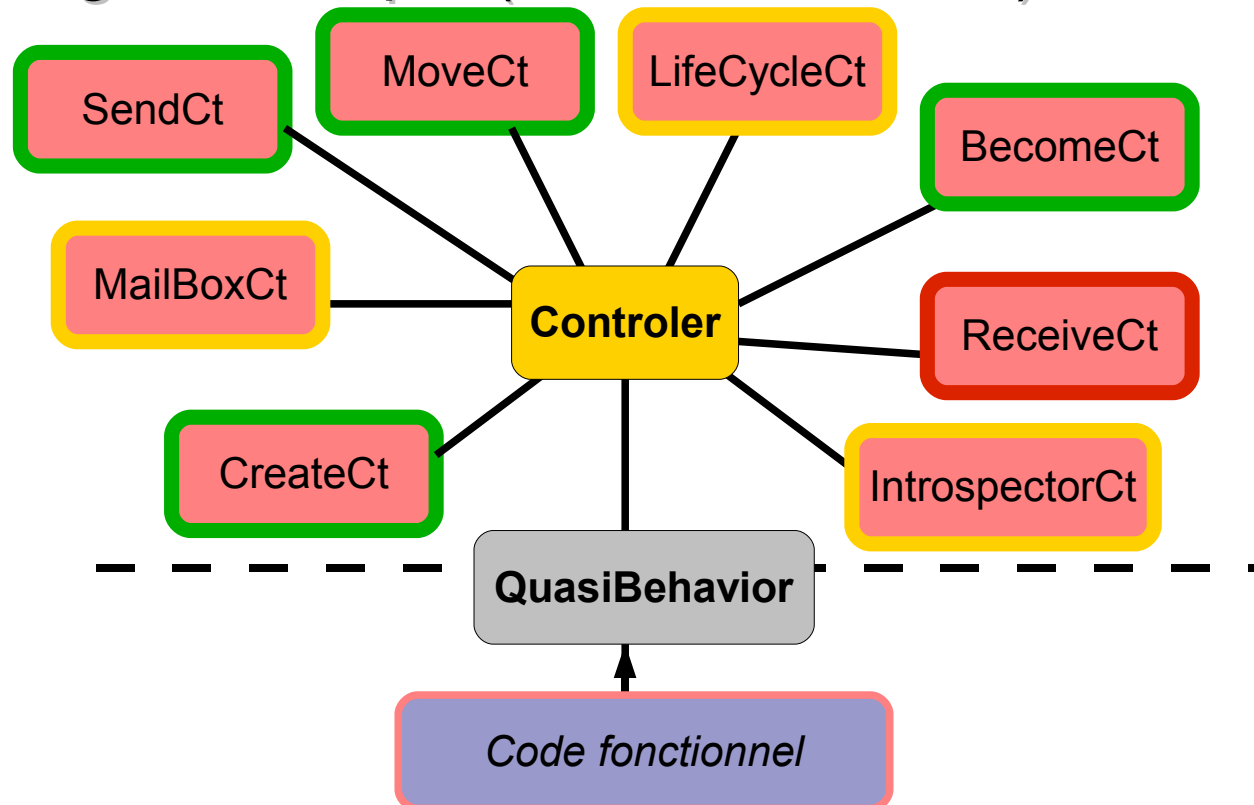
- Diversité des implémentations de SMA
  - Différents modèles
  - Différentes « composantes »
    - Cycle de vie
    - Perception environnement
    - Communication, etc.
  - Différentes plate-formes
  - Un outil pour implémenter différents types d'agents ?
- Possibilité de faire des agents « hybrides »
  - BDI mobiles, etc.
  - Réutilisation des  $\mu$ C

# Démarche (1)

- Objectifs :
  - Relâcher les contraintes du modèle d'AMA
  - Permettre la construction de différents modèles d'agents
    - Par assemblage de « capacités »
- Méthodologie : abstraction du modèle
  - Définition d'un **style d'architecture** d'agent :
    - Organisation en étoile autour d'un connecteur unique
    - Séparation des codes fonctionnels / non fonctionnels
    - Utilisation de composants à grain fin
    - Unicité des services (possibilité d'employer des  $\mu$ C composites)

## Démarche (2)

- L'adaptation n'est plus une propriété intrinsèque
  - On peut construire des agents non adaptables
- 3 catégories de  $\mu C$  (contrôle d'accès)

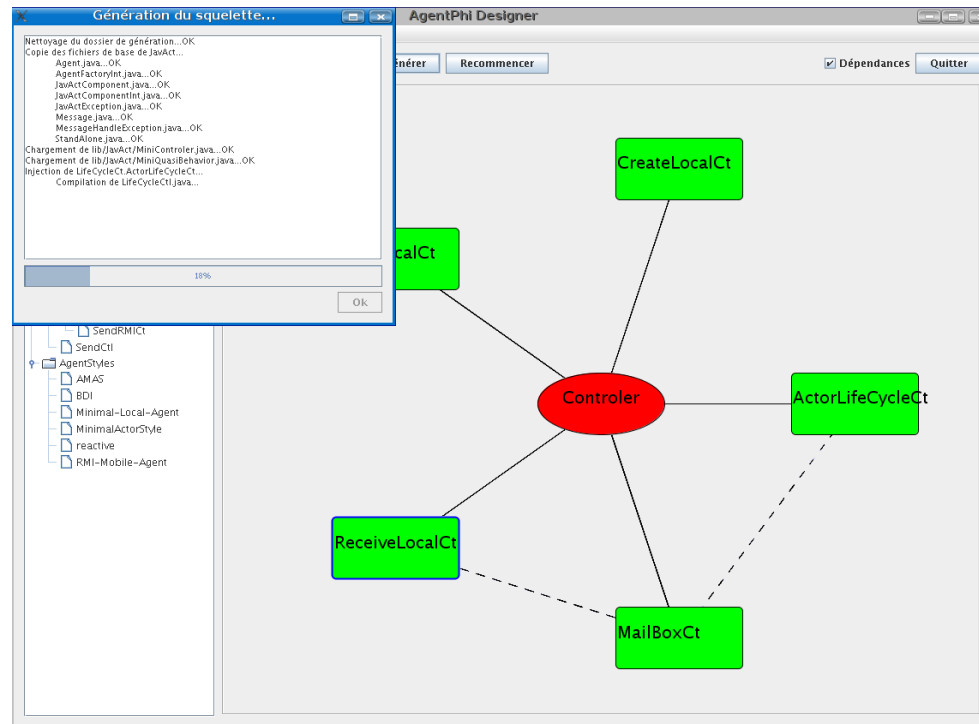


# Implication sur la conception

- Outil d'assemblage des composants
  - Description via un langage dédié (par ex. ADL)
  - Besoin / outils de validation
    - Assemblages de  $\mu$ C
    - Propriétés de l'architecture (cohérence de l'assemblage...)
  - Génération des squelettes de code de l'architecture d'agent

# Agent<sup>φ</sup>

- Prototype : Agent<sup>φ</sup> Designer
  - Modélisation / assemblage graphique
  - Vérifications simples
  - Génération de squelettes de code Java
- cf. Démo



# Conclusion

- **Modèle simple**
  - Assemblage de  $\mu$ C réutilisables
  - Possibilités d'intégration d'outils puissants (ADL+outillage)
- **Prototype développé en Java**
  - Usage simple, intégration avec les outils existants (plugin)
  - Code généré efficace
    - Faible taille, très bonnes performances à l'exécution
  - Bibliothèque de modèles d'agents et de  $\mu$ C
- **Travaux connexes (agents / composants)**
  - MALEVA, Magique, MAST, MaDcAr
  - Pas d'adaptation des services non-fonctionnels
  - Niveau de granularité (-fin, -précis)

# V. Conclusion & Perspectives

# Conclusion

- Contribution à la simplification du développement d'applications réparties flexibles
  - Architectures, patron de conception, outils logiciels
- Approche originale – combinaison de technologies
  - Agent mobile
  - Composants logiciels
  - Systèmes P2P
- Résultats probants
  - Simplicité d'utilisation (validation par les projets d'étudiants)
  - Forte réutilisation
  - Bonnes performances à l'exécution

# Perspectives & Questions ouvertes

- Intégration d'outils de description d'architectures dynamiques et outillage spécifique
- Ajout de contraintes dans les méta-données des  $\mu\text{C}$ 
  - Par exemple en OCL pour implémenter des contrats
- Expérimentations – projet IRIT
  - Intérêt du travail pour la communauté SMA ?
- Déploiement de composants standards ?
- Sémantique des modèles obtenus ? ( $\neq$  des acteurs)
- Changement dynamique de modèle ? (besoin ?)

# Questions ?