

# Architecture réflexive pour le contrôle de robots modulaires

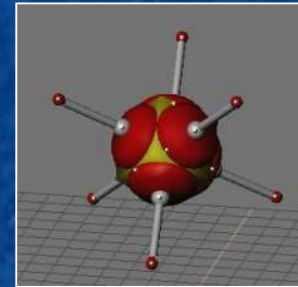
Jacques Malenfant\* \*\* et Simon Denier\*\*

\*LIP6, UPMC – CNRS (UMR 7606)

\*\*Université de Bretagne Sud - VALORIA

# Le projet MAAM

- Molécule := Atome ou (Atome + Molécule)
- Un atome : un module autonome
  - six pattes motorisées
  - mécanisme de connexion à chaque extrémité
- Une molécule : un robot modulaire (définition)
  - assemblage 2D ou 3D d'atomes
  - forme = fonction : déplacement tout-terrain, transport d'objets
  - reconfiguration : changement de forme pour s'adapter à l'environnement, à la tâche courante...



# Problématique de l'adaptation logicielle

- Implications de la reconfiguration
  - planifie et réalise le changement de forme
  - marque le passage d'une tâche à une autre, donc un changement du contrôle logiciel effectuant la tâche
- Besoin : adaptation dynamique du contrôle logiciel
  - comment ?
  - quand ?
  - efficacité ?

# Plan

- Préliminaires
  - Robotique modulaire : définition et problématique
  - Architectures de contrôle en IA
- Notre architecture logicielle
  - Framework réactif
  - Exécutif synchrone
  - Adaptation
  - Architecture réflexive
- Conclusion

# Architecture de contrôle en IA

- Rôle
  - définir les composants élémentaires du système
  - en organiser l'exécution
  - pour assurer la bonne marche du robot à court terme et/ou à long terme
- Composants pour :
  - perception
  - action
  - délibération

# Architecture réactive

- Ensemble de comportements indépendants et réactifs



- bonne réactivité aux opportunités temps réel ( « réflexe » )
- mais : séquences d'actions non optimales voire chaotiques, piège des « minima locaux »

# Architectures délibérative, hybride

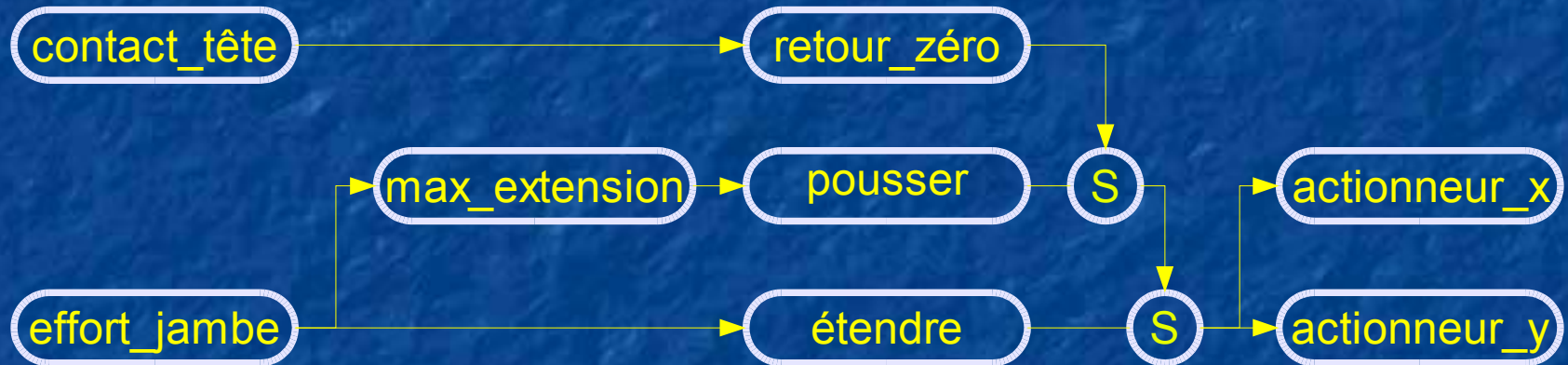
- Architecture délibérative
  - techniques de l'IA « forte » : modèle global
  - cycles : intégration des perceptions dans le modèle, puis décision
  - planification globale pour optimiser les actions
  - mais : faible réactivité en temps réel, modélisation complexe
- Architecture hybride délibérative/réactive
  - un niveau réactif chargé du contrôle temps réel
  - un niveau délibératif :
    - surveille l'évolution de la mission et
    - « collabore » avec le niveau réactif pour le maintenir à des niveaux d'efficiency et d'efficacité optimaux

# Plan

- Préliminaires
  - Robotique modulaire : définition et problématique
  - Architectures de contrôle en IA
- Notre architecture logicielle
  - Framework réactif
  - Exécutif synchrone
  - Adaptation
  - Architecture réflexive
- Conclusion

# Modèle de conception : la subsomption

- Composition et empilement de modules réactifs



- les modules réagissent à et transmettent des signaux
- connecteurs pour l'inhibition ou la « suppression » de signaux

# Le framework réactif

- Objectifs
  - donner un cadre (la subsomption) pour la programmation de modules réactifs, dans un langage haut niveau (JAVA)
  - offrir la machine d'exécution réactive pour les modules
- Vision utilisateur
  - conception du schéma, programmation et inscription des modules
  - programmation des modules (par héritage) :
    - conditions sur les signaux
    - implémentation de la méthode abstraite `handle()`

# Implantation : programmation réactive synchrone

- **Modèle synchrone : notion de cycle réactif**
  - à chaque cycle, le système évalue les signaux reçus et lance les activités correspondantes
  - intérêts : déterminisme (retestabilité), vérification formelle
- **Solutions adoptées**
  - REJO, extension réactive synchrone pour JAVA : abandonné
  - notre propre exécutif synchrone minimal : retenu pour réaliser une intégration harmonieuse des objets concurrents asynchrones (délibération) et des objets réactifs synchrones (contrôle)

# Exécutif synchrone minimal

- Le package ActiveObject (objets actifs et réactifs)
  - un support minimal pour l'exécution d'objets actifs, intégrant une communication asynchrone
  - à la base, modèle d'exécution asynchrone (réagit à l'arrivée d'un événement)
  - extension : le modèle synchrone (réagit aux événements lors d'un cycle)
- Intérêt de la communication asynchrone
  - environnement hétérogène mais transparent : un objet asynchrone peut communiquer avec un objet synchrone et vice-versa

# Exécutif GALS, synchronisation

- GALS : Globalement Asynchrone, Localement Synchrone
  - préserver l'approche synchrone dans un système réparti en maîtrisant l'asynchronisme inhérent
  - possible grâce à la communication asynchrone
- Synchronisation dans un système réparti GALS
  - assurer une synchronisation entre deux objets synchrones sans casser leurs contraintes d'exécution (pas de blocage)
  - notre solution : attente active sur des futurs
  - avantage : utilisation transparente des futurs entre objets concurrents asynchrones et objets réactifs synchrones

# Implantation : ordonnancement des modules

- Métaphore du schéma réactif : graphe partiellement ordonné
  - modules = sommets = ensemble à ordonner
  - signaux = arcs orientés = contraintes d'ordres
  - contrainte : ordre partiel entre les modules (pas de cycle)
- Ordonnancement statique par tri topologique

# Plan

- Préliminaires
  - Robotique modulaire : définition et problématique
  - Architectures de contrôle en IA
- Notre architecture logicielle
  - Framework réactif
  - Exécutif synchrone
  - Adaptation
  - Architecture réflexive
- Conclusion

# Scénarios d'adaptation

- Changement de tâche
  - reconfiguration, détection d'une fin de tâche
  - changement de tout ou partie des modules réactifs
- Tolérance aux pannes
  - détection d'une défaillance capteur ou moteur
  - remplacement des modules concernés par des modules palliatifs
- Optimisation de paramètres
  - amélioration des performances d'un module (non étudié dans ce stage)

# Framework adaptatif

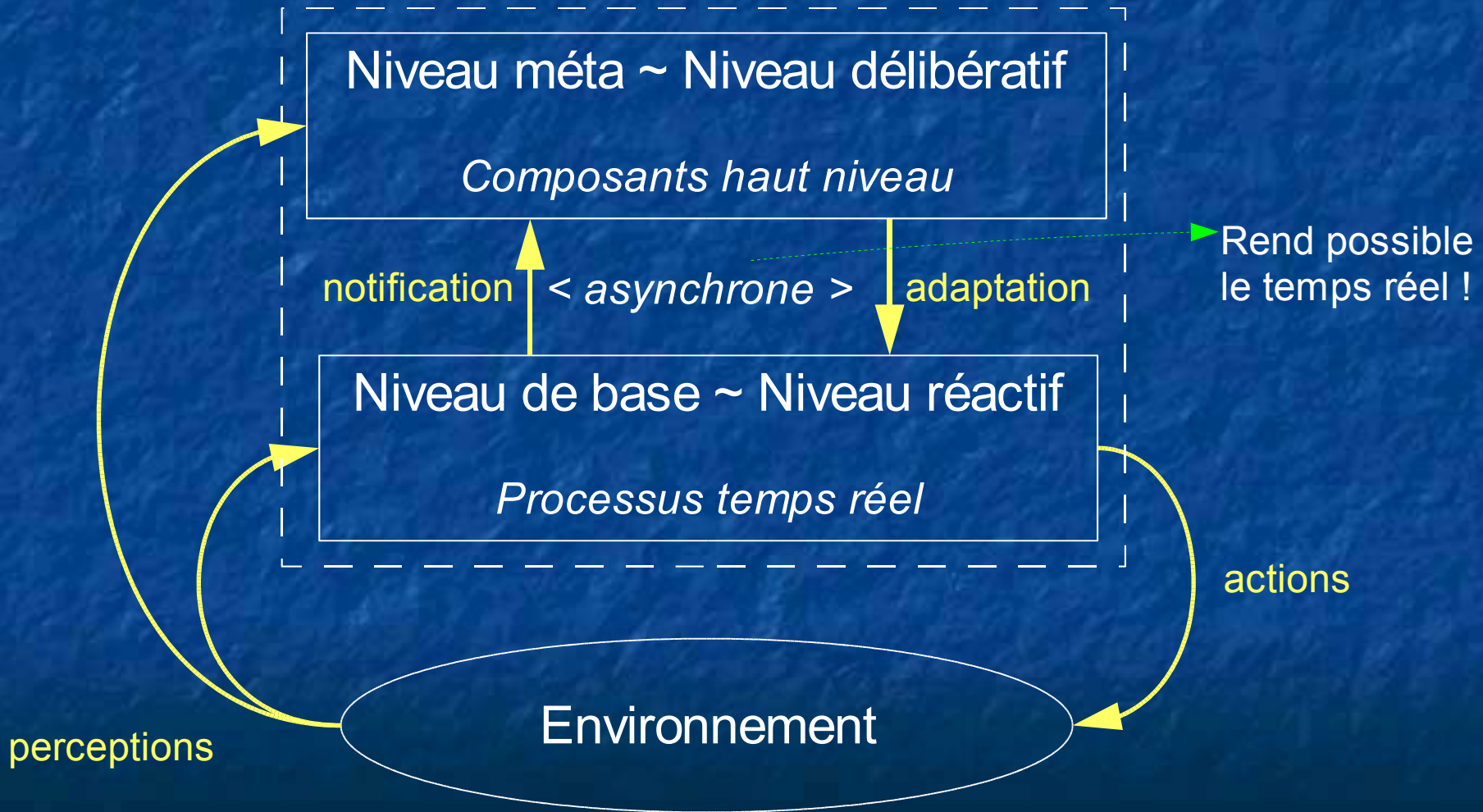
- Objectif : faciliter l'adaptation
- Mécanismes de modification dynamique
- Différents grains d'adaptation
  - tolérance aux pannes : grain = module réactif
  - changement de tâche : grain = ensemble de modules = « schéma » réactif

# Notre modèle réflexif :

## ARM

- Réflexivité : capacité d'un programme à se connaître et à se modifier dynamiquement
  - un niveau de base fait les calculs « normaux » de l'application
  - un méta-niveau fait des calculs sur le niveau de base pour l'adapter à de nouvelles conditions
- ARM (Asynchronous Reflection Model)
  - un noyau objet réflexif pour les systèmes répartis et réactifs
  - dissocier l'exécution du niveau de base de celle du méta-niveau via une communication asynchrone (ActiveObject)

# Architecture hybride réflexive



# Protocole d'adaptation

- Notifications du niveau de base vers le méta-niveau
  - à chaque cycle, émission de données sur l'état de l'atome (capteurs, actionneurs)
- Traitement des notifications au méta-niveau
  - intégration à la représentation de l'état courant
  - délibération sur l'état courant
- Envoi d'une requête d'adaptation si nécessaire
  - un appel sérialisé de méthode, appliqué par le niveau de base au cycle réactif suivant
  - avant ou après les activités réactives du cycle ?

# Conclusion

- Objectifs atteints
  - exécutif synchrone minimal type GALS
  - framework de contrôle réactif sur modèle haut niveau
  - intégration dans une plate-forme réflexive
  - développement de scénarios d'adaptation dynamique
- Expérimentations
  - expériences sur un schéma typique et complet
  - modèle de l'atome encore immature : un frein pour les expériences

# Architecture réflexive pour le contrôle des robots modulaires

Jacques Malenfant\* \*\* et Simon Denier\*\*

\* LIP6, UPMC – CNRS (UMR 7606)

\*\*Université de Bretagne Sud - VALORIA

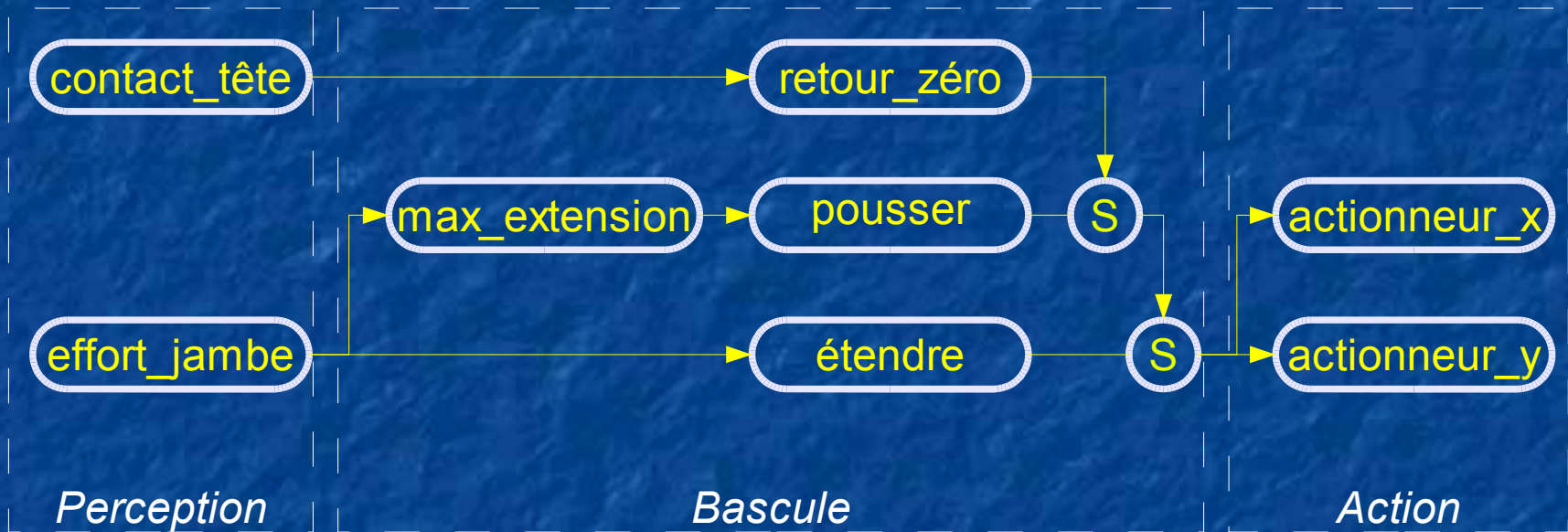
# Perspectives sur le framework réactif

- Validation du framework réactif
  - modèle simple, flexible mais relativement haut niveau
  - validation des performances (réactivité temps réel)
  - portage sur des technologies embarquées
- Développement d'un modèle de programmation de l'atome fondé sur la symétrie
  - un modèle symétrique permet l'usage de schémas génériques
  - problème : les référentiels mécaniques changent

# Perspectives sur le niveau délibératif

- Etude des performances liées à la communication en terme de cohérence/pertinence des informations et des interventions (délai)
- Stabilisation des APIs pour la programmation délibérative

# Exemple avec schémas réactifs



- Composition, empilement de schémas (ordre partiel)
  - préordonnancement interne à un schéma

# Vision utilisateur

- Construire le schéma réactif typé subsomption
- Créer les modules réactifs du schéma
  - donner des conditions sur les signaux (présence, absence)
  - programmer les réactions à ces conditions

```
if ( conditionsAreChecked() )  
    response = handle() ;  
generate( mySignal(response) ) ;
```

- Les inscrire pour l'exécution par la machine réactive
- Ajouter des connecteurs si nécessaire

# Niveau délibératif

- Rôle :
  - surveillance et adaptation du niveau réactif
  - collaboration avec les autres atomes (non étudié dans ce stage)
- Organisation réflexive à l'échelle moléculaire

