

DÉVELOPPEMENT DE SMA

Cours 1b

Je veux développer mon application..

- Comme toujours, j'utilise Java/Python/C#/...
- Mais :
 - Est-ce que je ne reprogramme pas exactement la même chose que quelqu'un d'autre? (en moins bien)
 - Est-ce que je ne pourrais pas avoir un langage un peu plus « intuitif » pour décrire mon agent?
 - Je veux utiliser java, mais maintenant, concrètement, je programme quoi?

Je regarde les plateformes

- Choix... important!
 - http://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software
- Nombreux critères:
 - Applications possibles
 - Langage de programmation
 - License

Développement de SMA

- Les plateformes génériques
 - Les langages
 - Normes et extensions
 - Plateformes utilisant un langage générique
 - JADE
 - MadKit
- Plateformes de simulation
 - RePast S
 - NetLogo
 - GAMA
 - FLAME
 - Synthèse
- Les plateformes pour d'autres applications
 - Pour la 3d: Massive
 - Pour la finance: MetaTrader, ATOM
- Plateformes liées à un modèle
 - JEDI
- Les outils d'aide au développement
 - Plateforme distribuée / collaborative
 - OpenMole
- GAMA: un exemple de développement de SMA
 - La plateforme GAMA
 - Application à un exemple
 - Extension de la plateforme

Je veux quand même tout programmer...

- Langage: Java / C++ / C# / ...
- Concrètement, un agent est...
 - Une application
 - Vraiment autonome
 - Distribution possible
 - Un thread
 - Permet des fonctionnements parallèles
 - Facilite l'administration
 - Un objet
 - Facilite la communication et la gestion des variables
 - Permet un contrôle complet

Je veux une solution très légère

- <http://sourceforge.net/projects/aglets/>
- Objectif: agent mobile
- Langage: Java
- License: IBM public license
- Identique à une applet mais en conservant son état (serialisé)
- Peut être transféré d'un *aglet host* à un autre
- **The aglet lifestyle**
 - **Created:** a brand new aglet is born -- its state is initialized, its main thread starts executing
 - **Cloned:** a twin aglet is born -- the current state of the original is duplicated in the clone
 - **Dispatched:** an aglet travels to a new host -- the state goes with it
 - **Retracted:** an aglet, previously dispatched, is brought back from a remote host -- its state comes back with it
 - **Deactivated:** an aglet is put to sleep -- its state is stored on a disk somewhere
 - **Activated:** a deactivated aglet is brought back to life -- its state is restored from disk
 - **Disposed of:** an aglet dies -- its state is lost forever J

Une plateforme légère respectant AGR pour simplifier l'organisation et les communications

- <http://www.madkit.org/>
- Plateforme très légère en Java
- Modèle AGR pour gérer la communication et les interactions

The screenshot displays the MadkitDesktop application interface. On the left is a file explorer showing a project structure with folders like 'autoload', 'scripts', and 'pythonfiles'. The main window is divided into several panes:

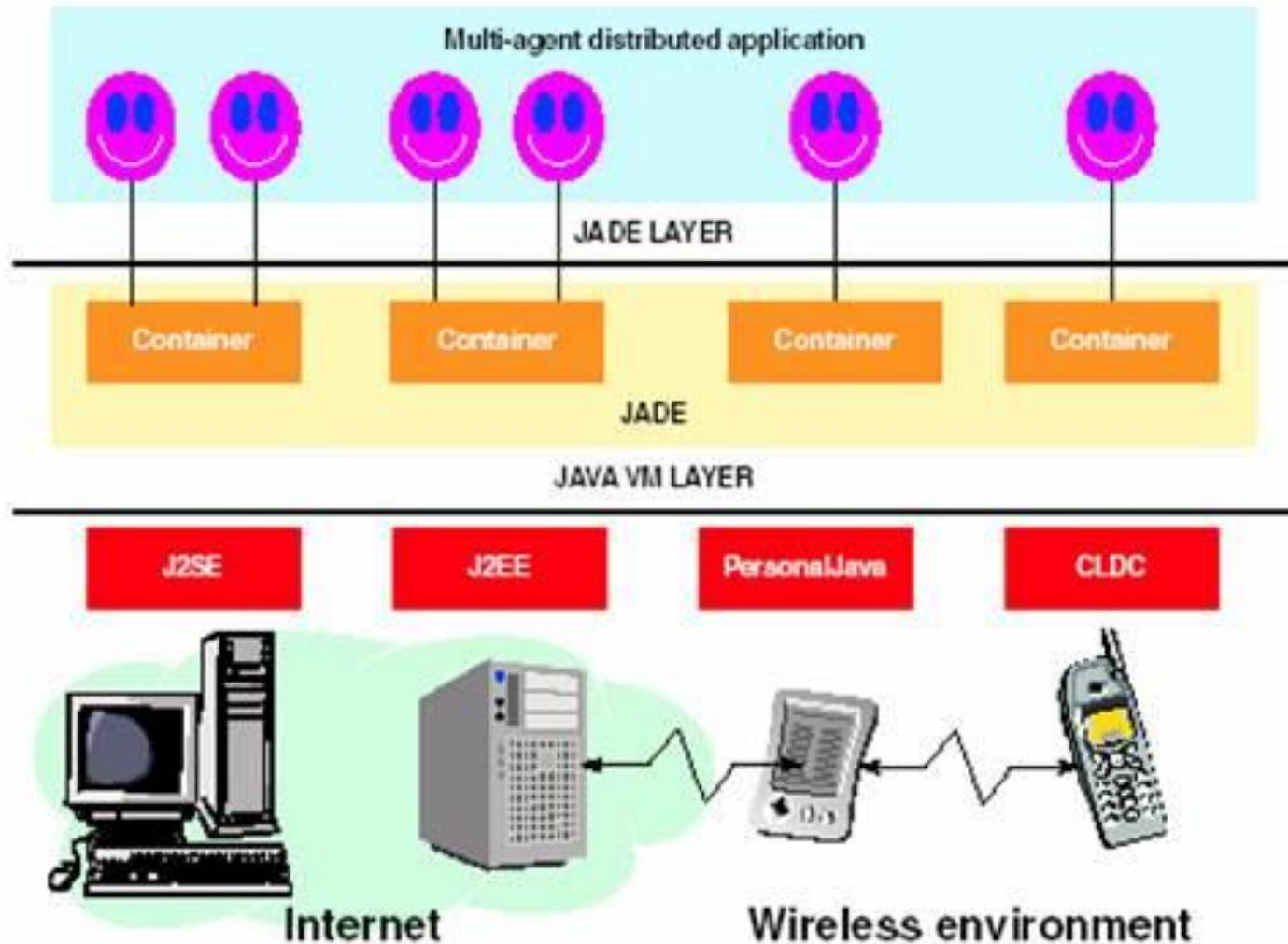
- GroupObserver-2:** A central pane showing a hierarchical view of agents and groups. It includes a 'public' group containing 'communications', 'jedit', 'system', 'ping-pong', 'python', and 'TERMITES'. Below this is a 'Messages' table with columns for 'Sender', 'Receiver', and 'Message Content'. A 'PythonAgent' window is also visible, showing a list of agents with their IDs and addresses.
- Code Editor (jEdit - PingPong.py):** Displays the source code for a Python agent. The code includes imports for 'Agent', 'StringMessage', and 'java.util'. It defines an 'activate()' method for group management and a 'live()' method for finding and interacting with other agents.
- Console/Activity Log:** Shows the execution output, including messages from the kernel and the PythonAgent, such as 'removing community : travel' and 'jEdit launched'.

The status bar at the bottom indicates the current environment is 'python', the CPU usage is 'Cp1252', and the memory usage is '15Mb/17Mb'.

Je veux une solution complète multi-plateforme: JADE

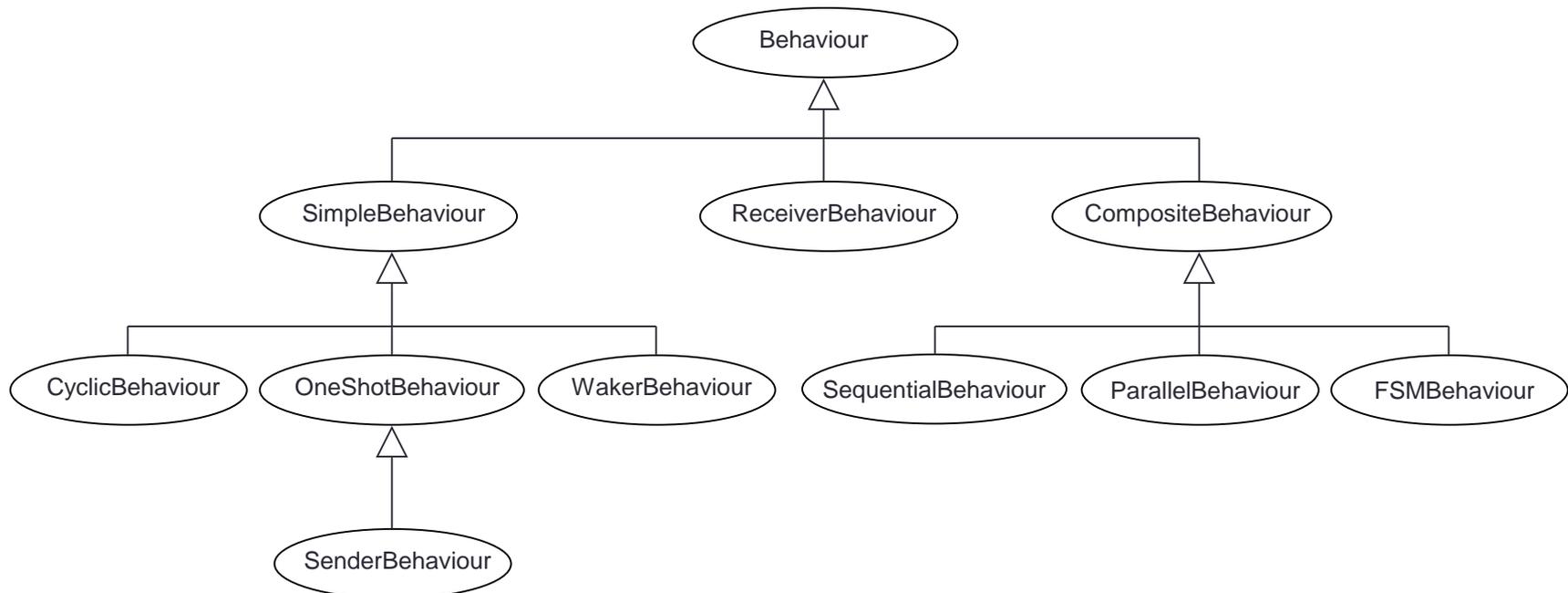
- <http://jade.tilab.com/> (Telecom Italia)
- Objectif: généraliste, middleware
- Langage: Java
- License: Open Source LGPL

Jade : agents mobiles



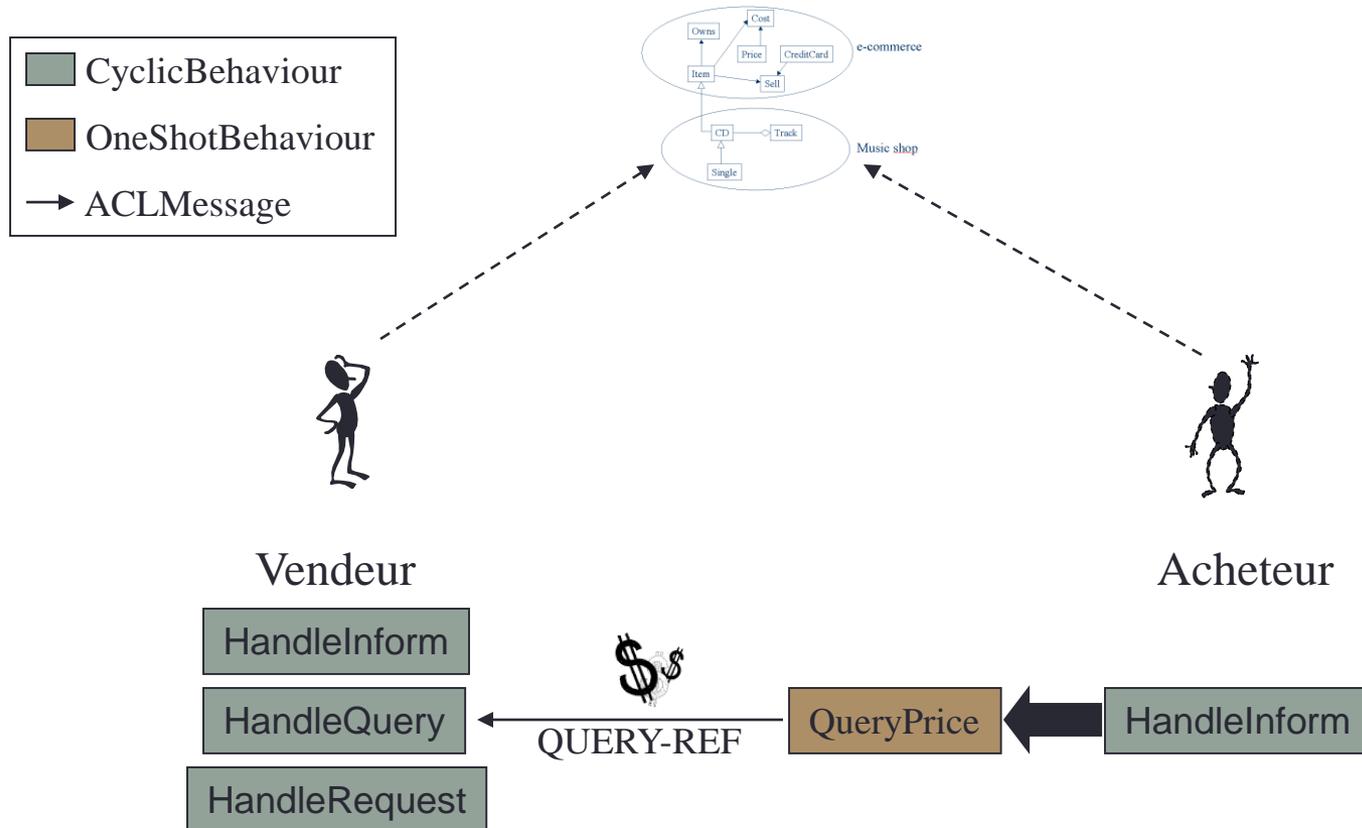
JADE: un agent...

- Hérite de la classe Agent
- Propose un/des services visibles dans un annuaire
- Dispose de plusieurs Behaviours, héritant de Behaviours ou de ses fils (one-shot, cycle, ...)
- Chaque agent et chaque Behaviour est un thread
- Envoie / reçoit des messages FIPA-ACL de façon transparente



Jade: exemple

[Pelissier 02]



Jade : Les principaux intérêts

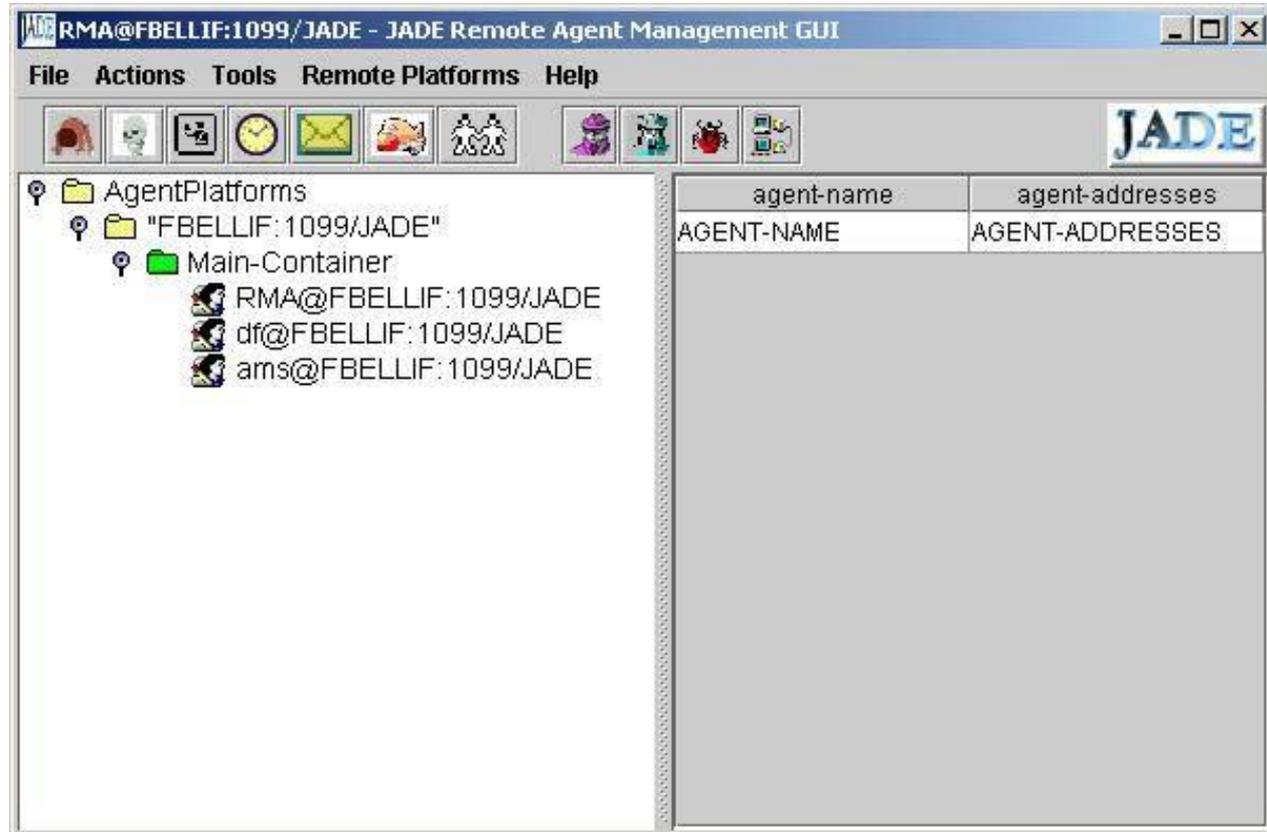
- Compatible FIPA
- Communauté importante
- Exécution distribuée
- Exécution concurrente des agents
- Communication transparente par message (ACL)
- Choix automatique de la meilleure méthode de transmission
- Notion de services
- Relativement facile à utiliser

Jade: Outils

- Jade GUI
- DF Agent GUI
- Dummy Agent
- Sniffer Agent
- Introspector Agent

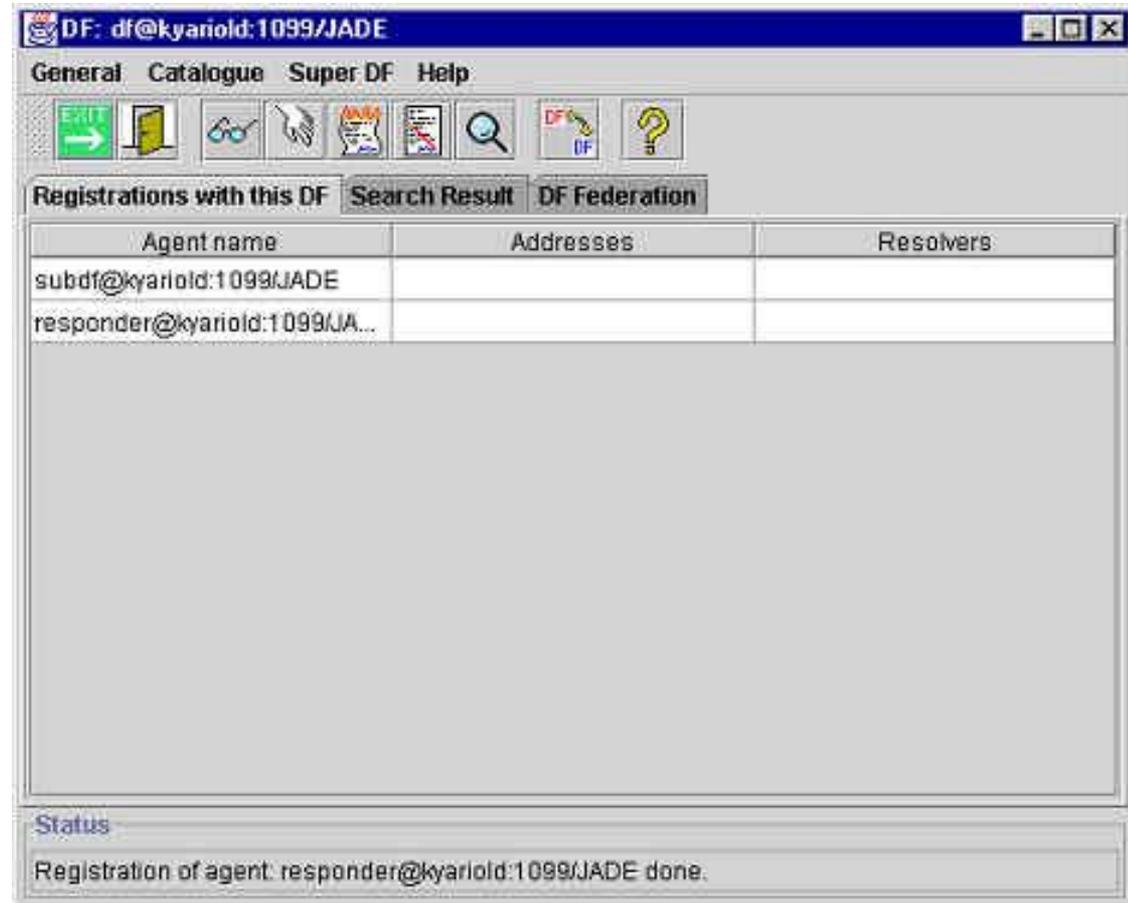
Jade GUI

- Contrôler les agents
 - Créer
 - Tuer
 - Suspendre
 - ...
- Démarrer les autres outils



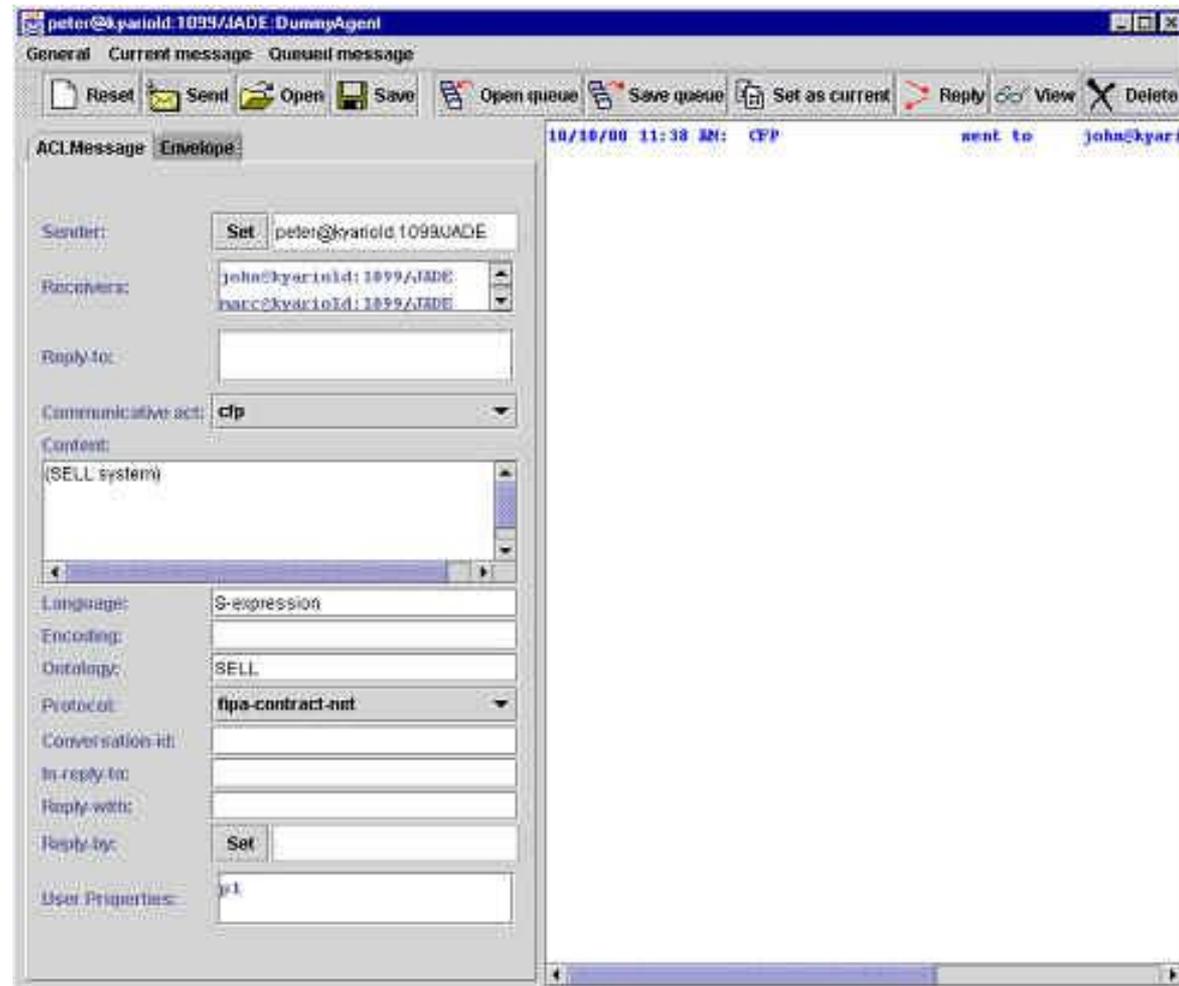
DF Agent GUI

- Inspecter les Yellow Pages (services enregistrés)



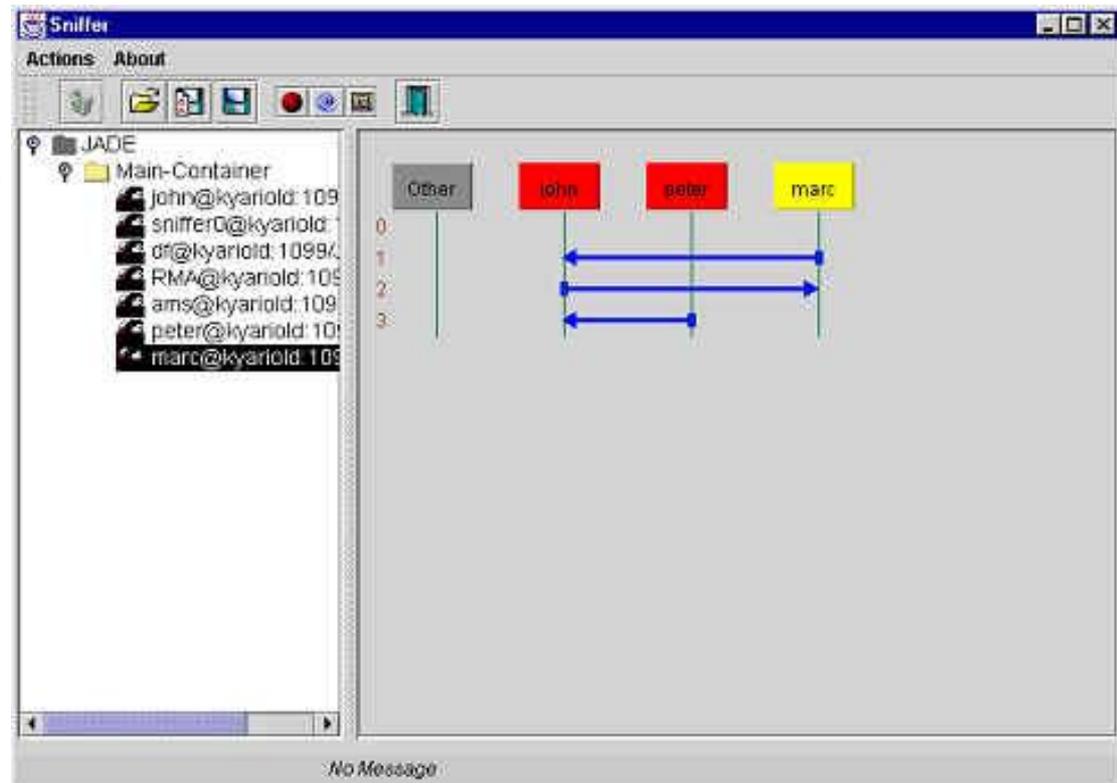
Dummy Agent

- Système de tests
- Envoyer et recevoir des messages ACL



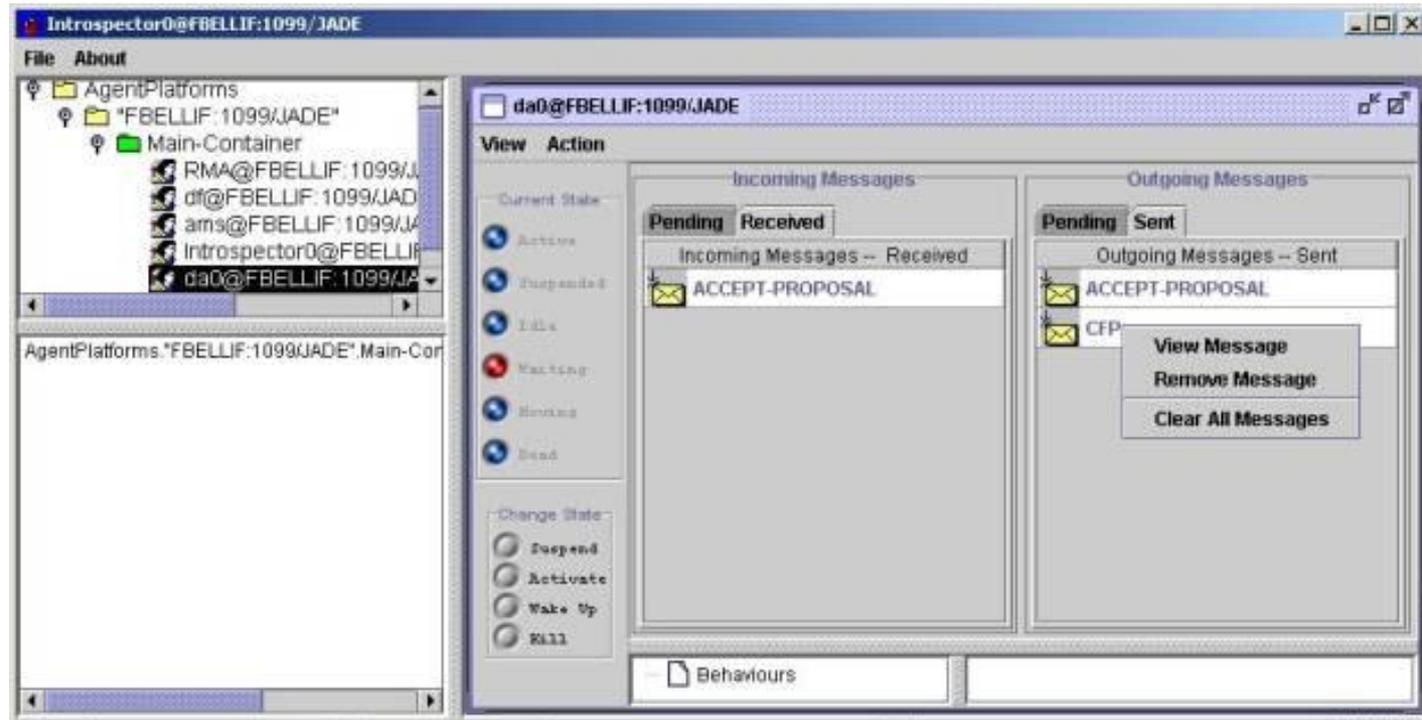
Sniffer Agent

- Surveillance des échanges de messages dans une plateforme



Introspector Agent

- Surveiller état (cycle de vie) d'un agent
- Ainsi que les messages reçus ou émis par cet agent



Développement de SMA

- Les plateformes génériques
 - Les langages
 - Normes et extensions
 - Plateformes utilisant un langage générique
 - JADE
 - MadKit
- Plateformes de simulation
 - RePast S
 - NetLogo
 - GAMA
 - FLAME
 - Synthèse
- Les plateformes pour d'autres applications
 - Pour la 3d: Massive
 - Pour la finance: MetaTrader, ATOM
- Plateformes liées à un modèle
 - JEDI
- Les outils d'aide au développement
 - Plateforme distribuée / collaborative
 - OpenMole
- GAMA: un exemple de développement de SMA
 - La plateforme GAMA
 - Application à un exemple
 - Extension de la plateforme

Je veux faire une simulation (agents synchrones, pas de problème d'ouverture, contrôle complet), avec un maximum de liberté: Repast

- <http://repast.sourceforge.net/>
- Objectif: simulations multi-agent
- Langage:
 - Java, C++, C#, Objective C (RePast S)
 - Java, Objective C (RePast J)
- License: BSD

RePast 3

- A l'origine, un timer et des outils de gestion du temps, des paramètres et des variables (RePast J)



RePast Symphony

- Récemment, une interface de modélisation (RePast S)

The screenshot shows the Eclipse IDE with the RePast S modeling interface. The main workspace displays a state transition diagram for the `GasNode.agent` model. The diagram includes the following elements:

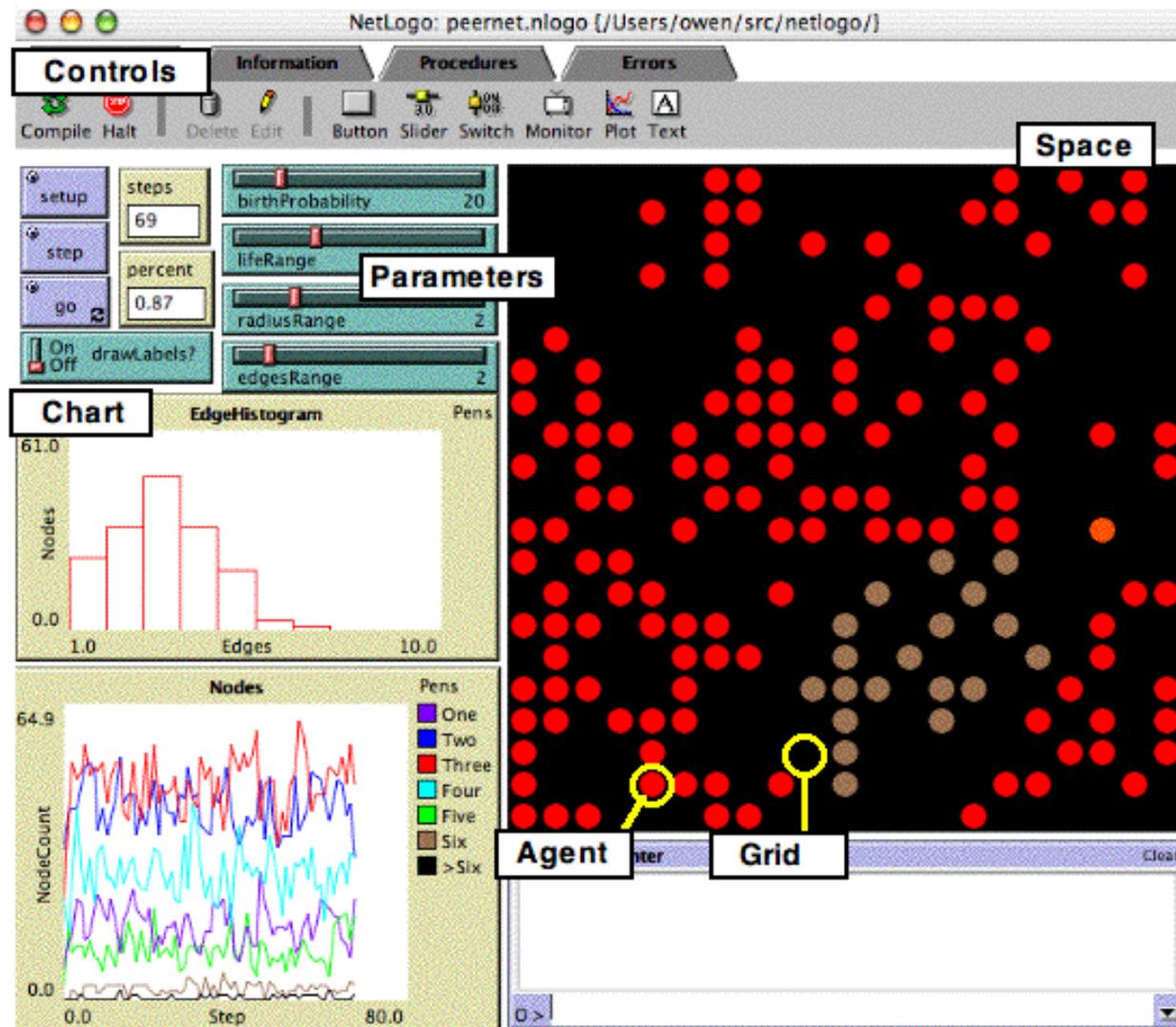
- Start Node:** A triangle labeled "Node pressure".
- Transition:** A downward arrow labeled "Watch for pressure change".
- Decision Node:** A diamond labeled "Evaluate pressure change criteria".
- Transitions from Decision:**
 - A "true" path leading to a task node "Change Pressure".
 - A "false" path leading to a task node "Average measurements".
- Task Node:** A rectangle labeled "Change Pressure".
- End Node:** A downward-pointing T-shape.
- Start Node:** A triangle labeled "Measure Pressure".
- Transition:** A downward arrow labeled "Define temp variable".
- Task Node:** A rectangle labeled "Take several measurements".
- Loop:** A feedback arrow from the "Take several measurements" task back to itself, labeled "true".
- Transition:** A downward arrow labeled "false" leading to the "Average measurements" task node.
- Task Node:** A rectangle labeled "Average measurements".
- Transition:** A downward arrow labeled "Measure" leading to a final task node.
- End Node:** A downward-pointing T-shape.

The console window at the bottom displays the following table:

Property	Value
Step 1: Type in a Comment that Describes this Task	This is a task.
Step 2: Type in a Human-Readable Diagram Label for the Property	Average measurements
Step 3: Optionally Choose an Type from the Task List	Context Operation: Find a Context [Context: FindContext(String contextf
Step 4: Optionally Note the Example Task	Context context = FindContext("root/sub1")
Step 5: Type in Task Part 1	measuredPressure = x / numTimesMeasured
Step 6: Optionally Type in Task Part 2	
Step 7: Optionally Type in Task Part 3	

Je veux programmer très vite un prototype: NetLogo

- <http://ccl.northwestern.edu/netlogo/>
- Objectif: Simulation
- Langage: NetLogo
- License: non Open Source <5
- OpenSource pour la futur v5

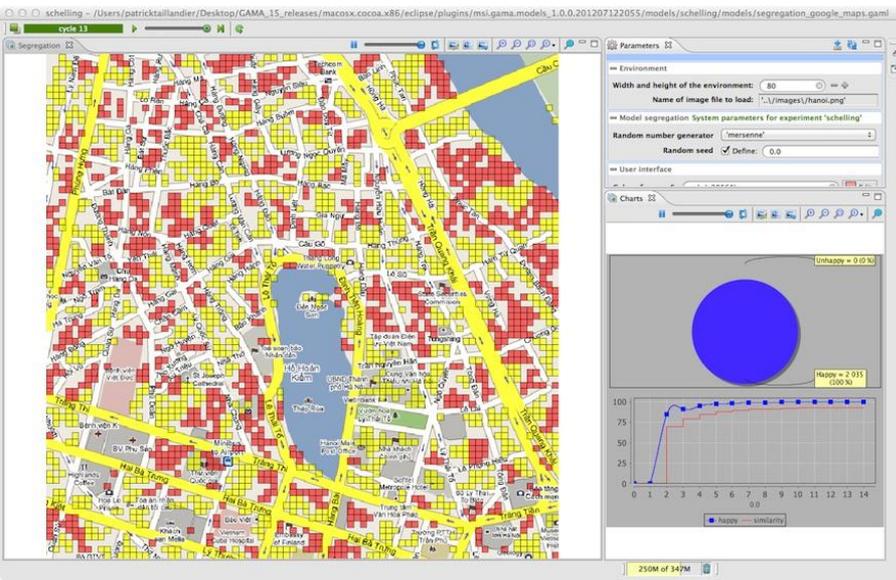


NetLogo: exemple

- **to setup**
 - **clear-all** ;; clear the world
 - **create-turtles 100** ;; create 100 new turtles with random heading
 - **ask turtles**
 - [**set color red** ;; turn them red
 - **forward 50**] ;; make them move 50 steps forward
 - **ask patches**
 - [**if (pxcor > 0)** ;; patches with pxcor greater than 0
 - [**set pcolor green**] ;; turn green
- **end**

Je veux programmer une simulation avec un langage dédié pour être plus efficace, tout en gardant une simulation rapide et la possibilité de programmer en java: GAMA

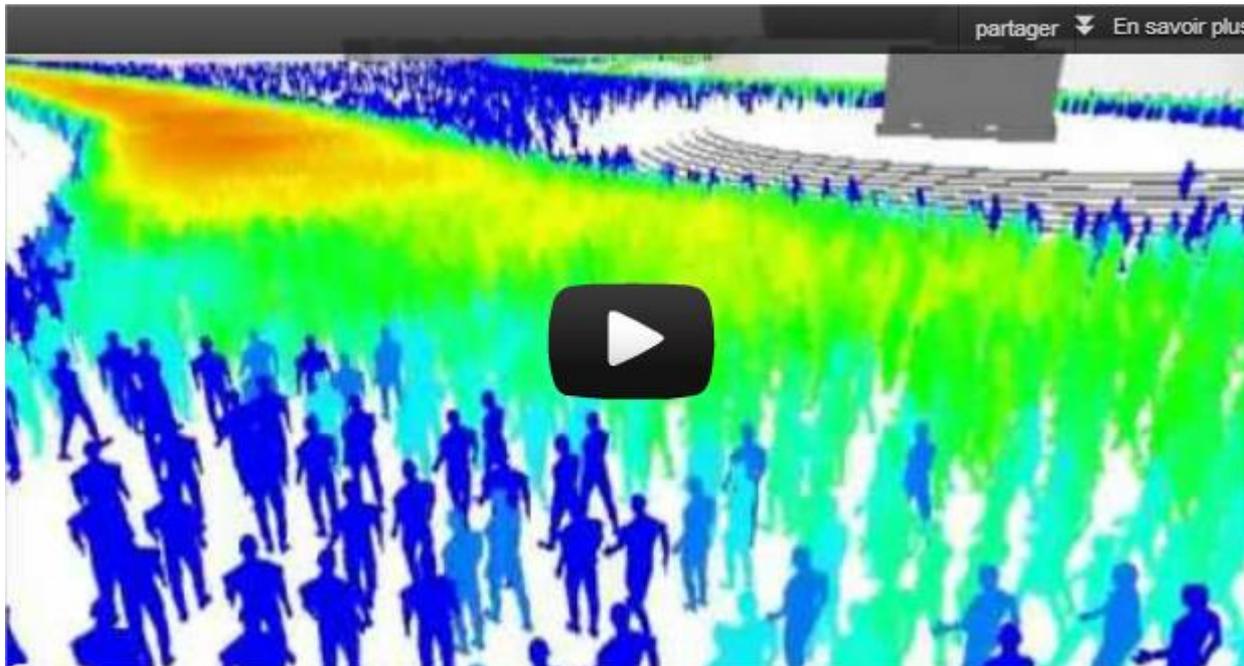
- <http://code.google.com/p/gama-platform/>
- Objectif: simulation
- Langage: XML (v1.3), GamL et Java (v1.4+)
- License: GPL



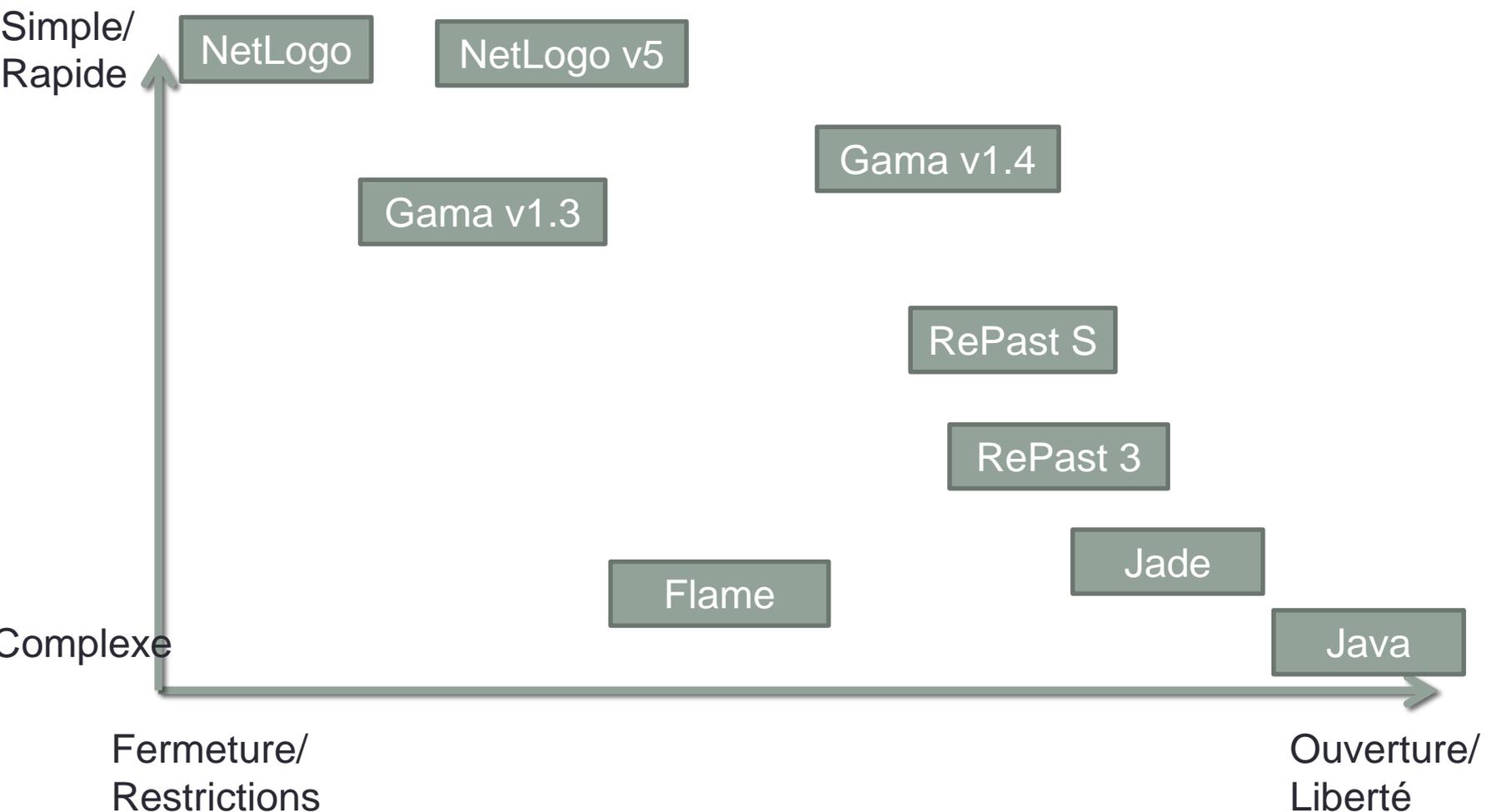
©2007-2012 IRD UMMISCO & Partners
<http://gama-platform.googlecode.com>

J'ai vraiment beaucoup d'agents très simple que je veux distribuer sur GPU: FLAME / FLAMEGPU

- <http://www.flamegpu.com/index.php>
- Contraint des diagramme d'état
- Permet le calcul sur GPU



Une plateforme simple ou générique?



Développement de SMA

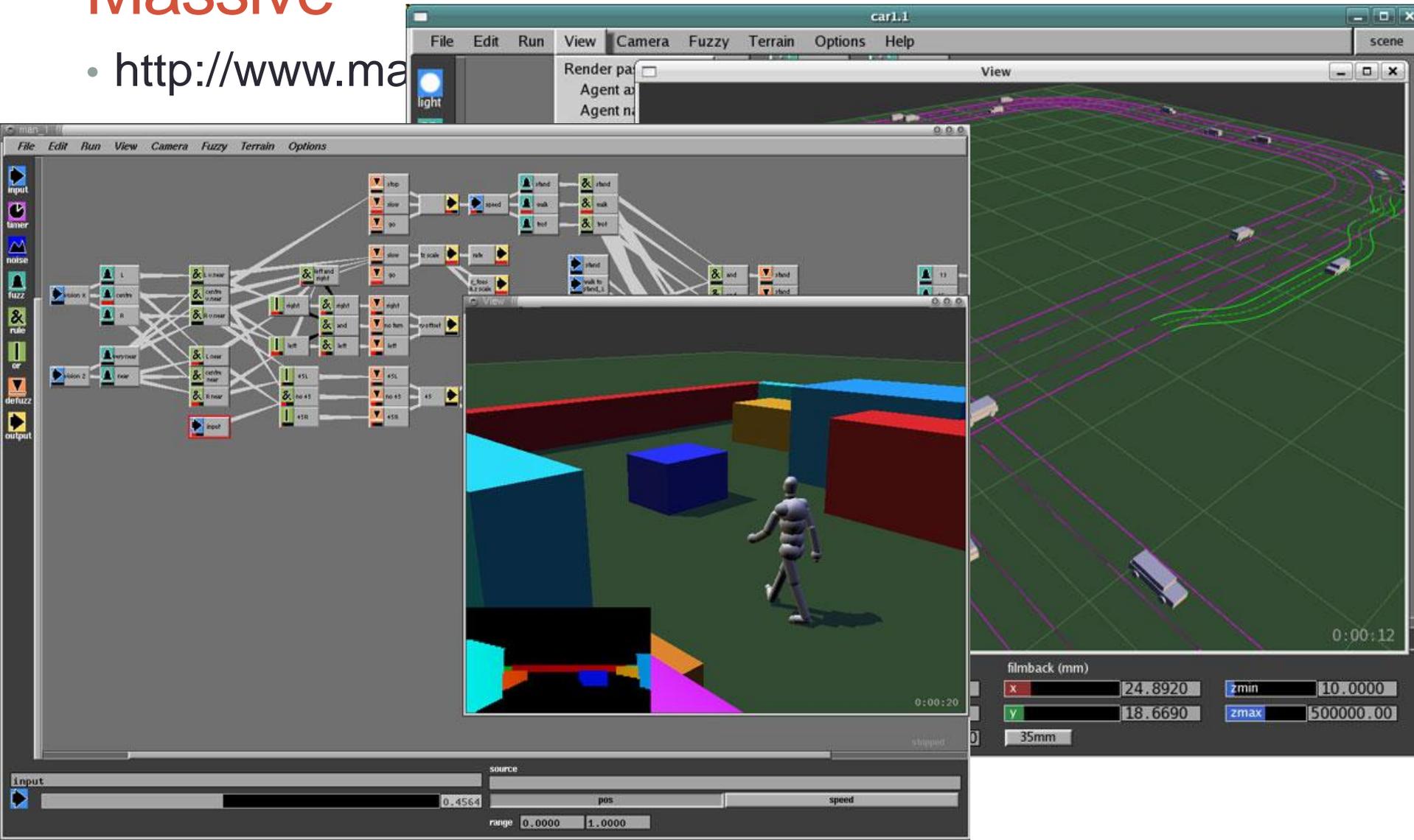
- Les plateformes génériques
 - Les langages
 - Normes et extensions
 - Plateformes utilisant un langage générique
 - JADE
 - MadKit
- Plateformes de simulation
 - RePast S
 - NetLogo
 - GAMA
 - FLAME
 - Synthèse
- Les plateformes pour d'autres applications
 - Pour la 3d: Massive
 - Pour la finance: MetaTrader, ATOM
- Plateformes liées à un modèle
 - JEDI
- Les outils d'aide au développement
 - Plateforme distribuée / collaborative
 - OpenMole
- GAMA: un exemple de développement de SMA
 - La plateforme GAMA
 - Application à un exemple
 - Extension de la plateforme

Je veux programmer un agent financier...

- Réaliser de vrais agents traders: MetaTrader
 - Installation et compte gratuit: <http://global.fxdd.com/fr/>
 - Marché: FOREX
 - Langage spécifique
 - Et/ou DLL C++
 - Outils de tests et d'exploration
- Simuler un marché: ATOM
 - <http://atom.univ-lille1.fr/>
 - <http://atom.univ-lille1.fr/js/>
- Avoir un agent qui puisse agir aussi bien sur une simulation que sur le marché réel (avec compte réel ou non): MetaSim

Je veux concurrence Peter Jackson: Massive

- <http://www.massive.cc>



Je veux définir des interactions et non des actions: IODA

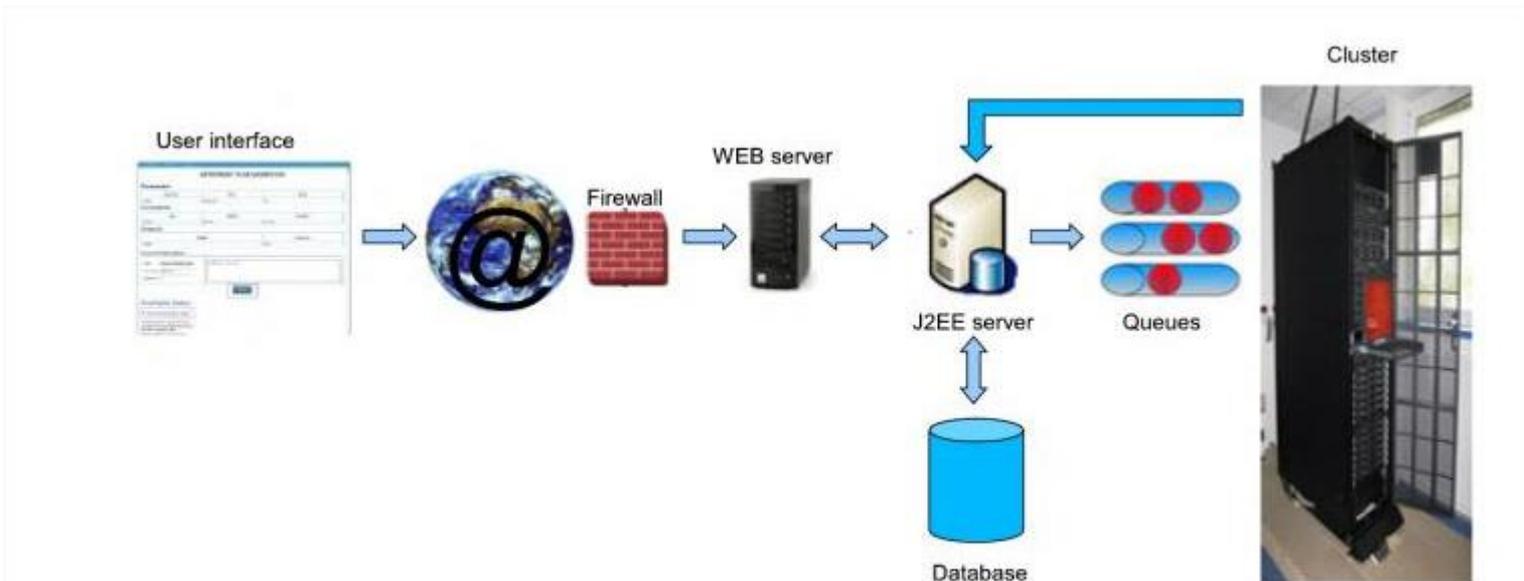
- API en Java
- Outils adaptés à la méthodologie
- <http://www.lifl.fr/SMAC/projects/ioda/index.php>

Développement de SMA

- Les plateformes génériques
 - Les langages
 - Normes et extensions
 - Plateformes utilisant un langage générique
 - JADE
 - MadKit
- Plateformes de simulation
 - RePast S
 - NetLogo
 - GAMA
 - FLAME
 - Synthèse
- Les plateformes pour d'autres applications
 - Pour la 3d: Massive
 - Pour la finance: MetaTrader, ATOM
- Plateformes liées à un modèle
 - JEDI
- Les outils d'aide au développement
 - Plateforme distribuée / collaborative
 - OpenMole
 - Yang
- GAMA: un exemple de développement de SMA
 - La plateforme GAMA
 - Application à un exemple
 - Extension de la plateforme

Distribution et coopération

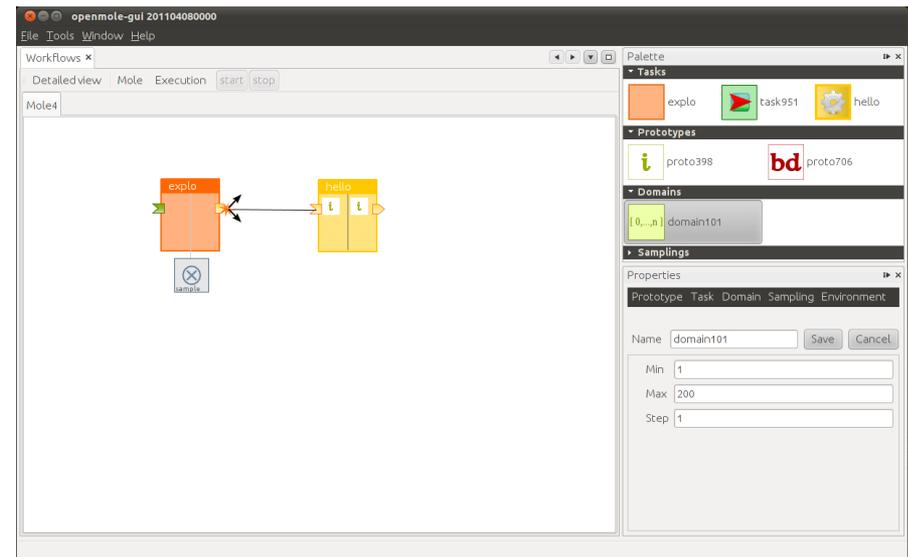
- 1^{ère} approche: accéder à un site web et déposer son code pour distribuer des simulation sur une grille
 - Ex: EPIS [Blanchart et al. 2011]
 - Accepte des projets NetLogo, Gama, ...
 - Un serveur doit être installé sur un ordinateur de la grille



OpenMole



- 2^{ème} approche: le programme de distribution est installé uniquement chez le développeur, il se charge de transférer ce qu'il faut sur la grille
- <http://www.openmole.org>
- [Reuillon et al. 2010]
- Avantage: transparent, gestion du workflow



RePast HPC

- 3^{ème} approche: distribuer la simulation
- Beaucoup plus complexe que distribuer les simulation
- Plateforme en version Beta pour clusters et supercalculateurs

