Outils et langages de programmation d'applications multi-agents

Olivier Boissier

Univ. Lyon, IMT Mines Saint-Etienne, Laboratoire Hubert Curien UMR CNRS 5516
Olivier.Boissier@emse.fr

PDIA – 6 Octobre 2017

Préambule

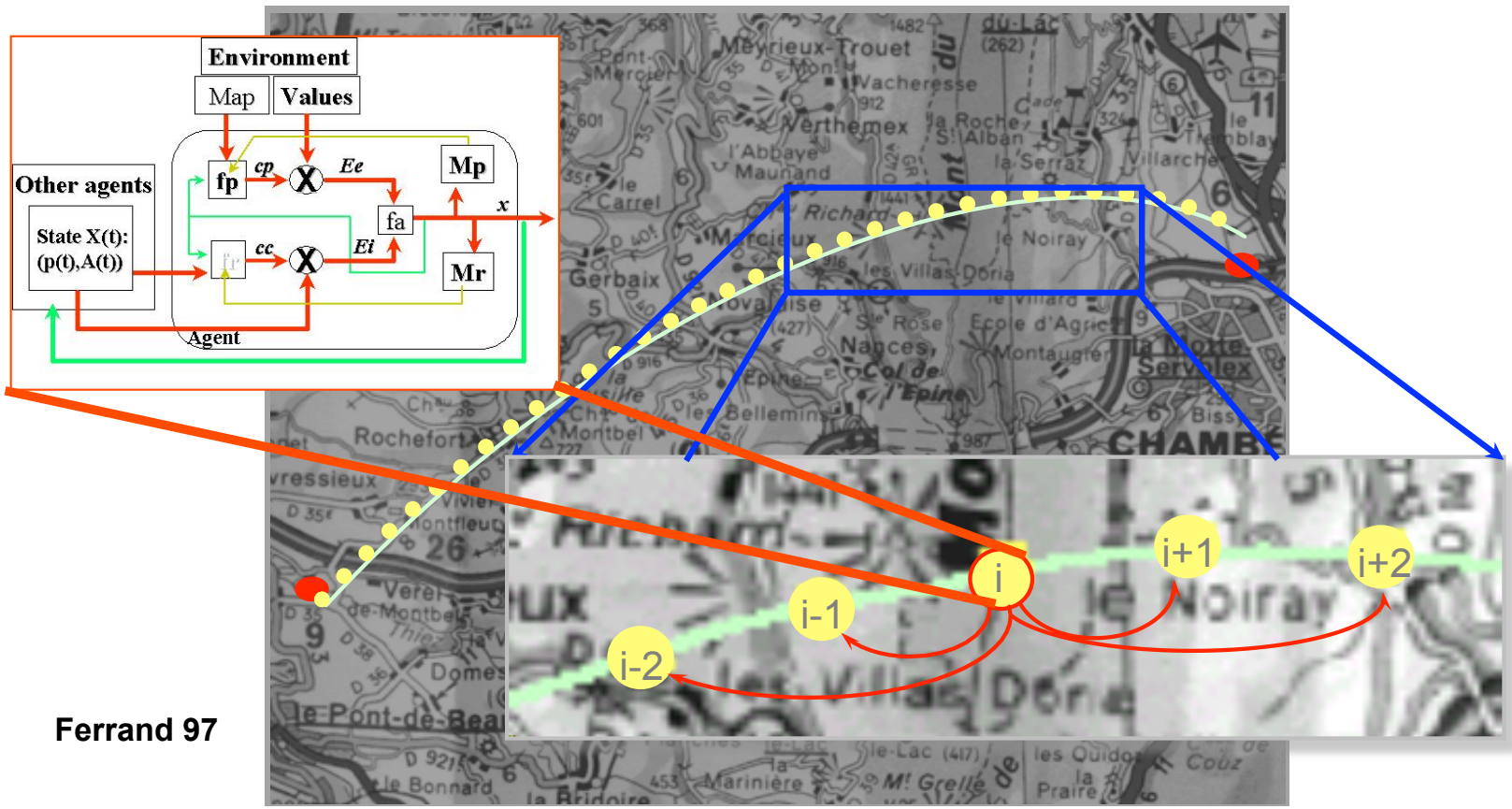
- Cette présentation a pour objectif de donner un panorama des directions actuelles dans le développement des outils et langages pour la programmation d'applications multi-agents
- Il ne s'agit pas :
  - d'un comparatif de ces outils et langages selon différents critères (e.g. performance, modèles)
  - d'un cours de programmation sur une plateforme particulière
- Sans pouvoir être exhaustif sur ces outils et langages, il s'agit d'inviter à aller plus loin et à étudier ces outils ainsi que les fondamentaux des systèmes multi-agents
- Centrée sur les systèmes multi-agents, les plateformes d'assistants personnels intelligents sont écartées de cette étude

Plan

Définition

Système multi-agent

Une organisation d'agents autonomes en interaction entre eux et avec un environnement partagé dans lequel ils sont situés

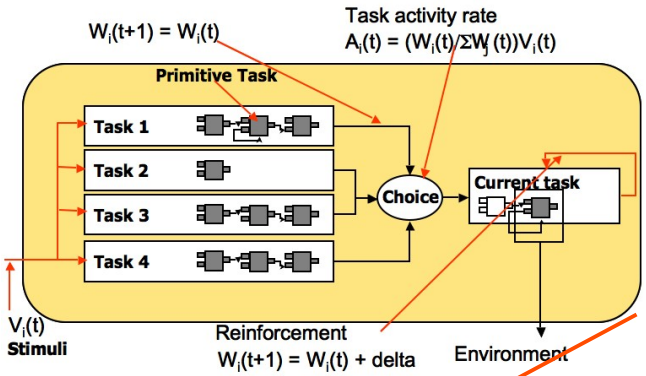


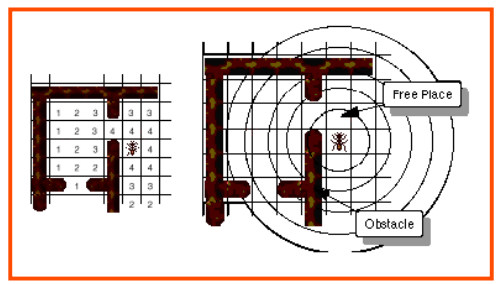
Ferrand 97

Définition

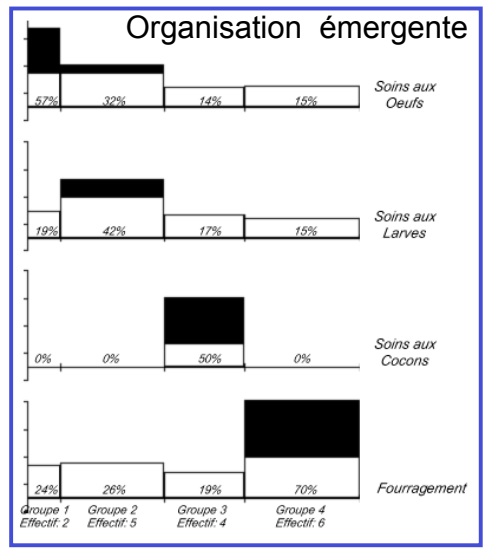
Système multi-agent

Une organisation d'agents autonomes en interaction entre eux et avec un environnement partagé dans lequel ils sont situés





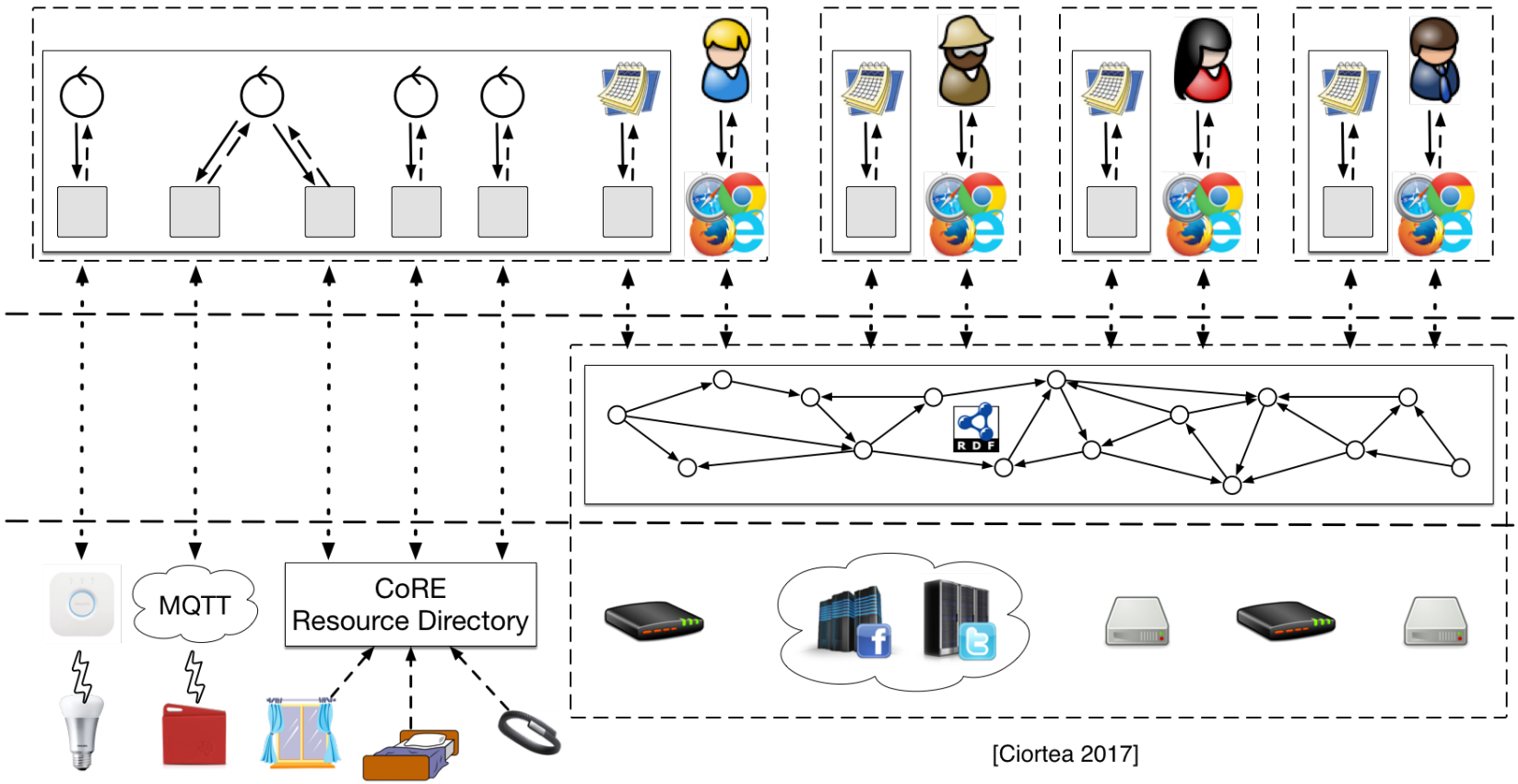
MANTA [Drogoul 93]



Définition

Système multi-agent

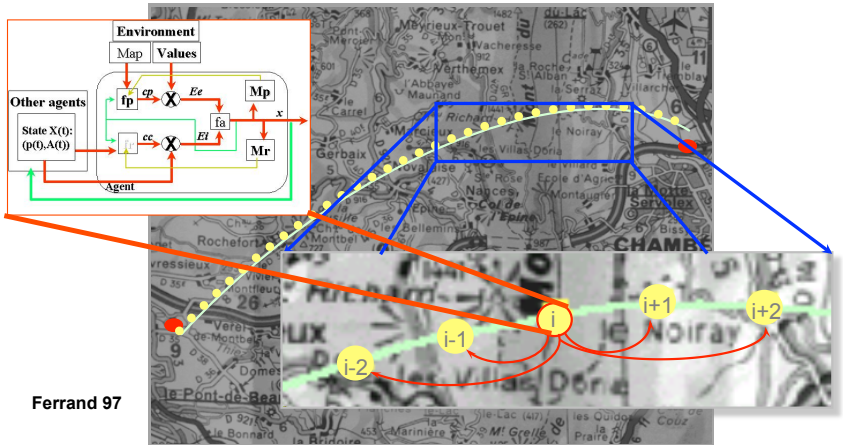
Une organisation d'agents autonomes en interaction entre eux et avec un environnement partagé dans lequel ils sont situés



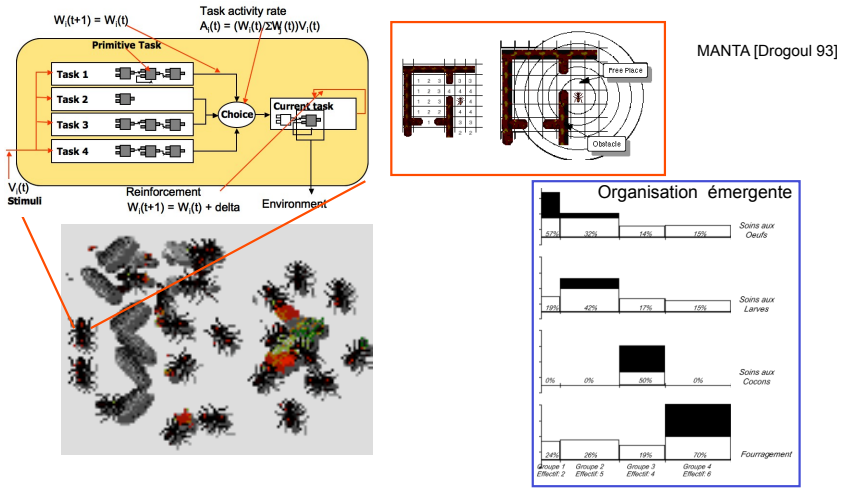
[Ciortea 2017]

Applications multi-agents

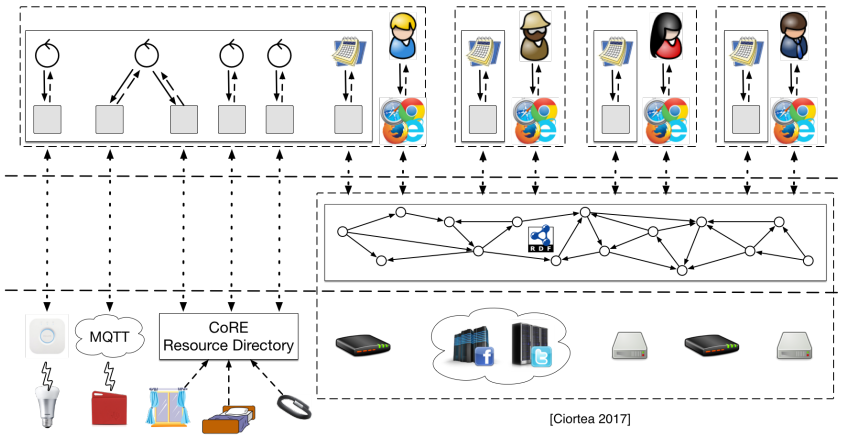
Système multi-agent



Résolution de problèmes multi-agent



Simulation multi-agent



[Ciortea 2017]

Intégration multi-agent

Caractéristiques :

- Distribution
- Centralisation impossible des décisions/du contrôle
- Environnements complexes, dynamiques

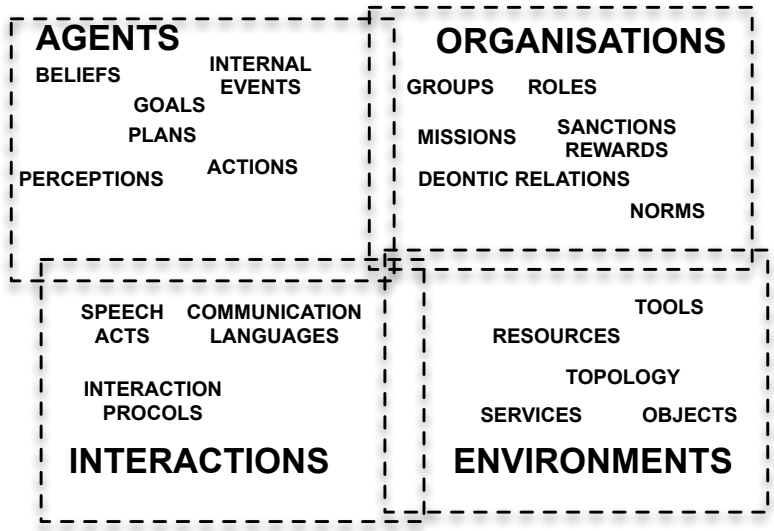
Comment programmer de telles applications?

Concepts et abstractions de développement

Système multi-agent

Abstractions et concepts pour :

- Définir l'architecture des entités autonomes qui agissent dans le système (Agent)
- Définir et structurer les ressources et traitements partagés entre les agents (Environnement)
- Définir les interactions entre entités (Interaction)
- Définir et structurer la coordination et la régulation au sein du collectif d'agents (Organisation)



Structuration possible selon VOWELS [Demazeau, 1995]

Nombreux concepts – Absence de consensus

Programmation de systèmes multi-agents

Méthodes d'ingénierie de systèmes multi-agents

- Existence de nombreuses méthodes [Bergenti et al., 2004]
- Couverture diverse des étapes du cycle de développement

Outils supports à l'ingéniérie :

- O-MASE AgentTool (pas de mise à jour depuis 2013), ZEUS (pas de mise à jour depuis 2013), Prometheus PDT, PASSI ToolKit, INGENIAS, AOM, ADELFE, ...

Outils supports à la programmation :

- Agent Toolkits, Agent Frameworks, Agent Programming Languages, Agent Platforms, Multi-Agent Platforms, Agent Development Environments

Plan

Champs d'action

Outils orientés simulation multi-agent

- Propriétés des applications cibles :
  - Représentations locales de différents points de vue, de décisions, de buts, de motivations, de comportements, etc
  - Interaction entre des stratégies, des comportements et des traitements locaux, éventuellement sous le contrôle de stratégies globales et communes
  - La solution est le résultat de l'interaction entre des processus locaux
  - Fonctionnement et évolution continus

- Caractéristiques intrinsèques :
  - Distribution logique et parfois physique
  - Décentralisation
  - Homogénéité fréquente des agents
  - Grand nombre d'agents
  - Importance de l'environnement, complexe, partagé et dynamique
  - Ouverture possible mais pas essentielle

Panorama des domaines d'application

Outils orientés simulation multi-agent

Table 2
Application domains covered by the ABMS tools.

ABMS scope or application domain	ABMS software tools
Cellular automata, Complex adaptive systems, Emergent complex phenomena in Biology/Medical sciences, Epidemiology, Artificial life (Evolutionary computation or genetic programming, Artificial intelligence, Neural networks, Robotics)	Agent Cell (2D/3D), AnyLogic (2D/3D), Ascape, BehaviourComposer (2D/3D), Breve (3D), BSim (2D/3D), DigiHive, Echo, EcoLab, FLAME, FLAME GPU (3D), Framsticks (2D/3D), GALATEA, GridABM, HLA_Agent, HLA_RePast, JAS, JCASim (1D/2D/3D), MASON (2D/3D), MASyV (2D/3D), Mathmatica® (Wolfram), Mesa, MIMOSE, MOBIDYC, PDES-MAS, Repast-J/Repast-3, Repast HPC, Repast Simphony (2D/3D), SEAS (2D/3D), SimAgent, SimBioSys, SOARS, Sugarscape, Swarm, TerraME, VisualBots, VSEit, Xholon (2D/3D)
Social & natural sciences, Dynamic computational Systems, Business, Marketing, Economics, Ecology, Healthcare, Planning & Scheduling, Enterprise and organizational behaviour, Traffic Situations (avoidance of traffic jams, light control, route choice)	Agent Factory, AgentScript, AgentSheets, AnyLogic (2D/3D), AOR Simulation, Ascape, BehaviourComposer (2D/3D), Brahms, Cormas, CybelePro, Echo, Envision, Eve, ExtendSim (2D/3D), FLAME, FlexSim, Framsticks (2D/3D), GALATEA, GAMA (2D/3D), GROWLab, HLA_Agent, HLA_RePast, IDEA, Insight Maker, JAMSIM, Janus, JAS, JAS-mine, jES, LSD (2D/3D), MASON (2D/3D), MASS, Mathmatica® (Wolfram), MATSim, Mimosa, MIMOSE, MOBIDYC, Modgen, NetLogo (2D/3D), Pandora, PDES-MAS, PS-I, Repast-J/Repast-3, Repast HPC, Repast Simphony (2D/3D), SeSAm, SimAgent, SimBioSys, SimEvents (MATLAB®), Simio (2D/3D), SimJr, SimSketch, Simul8, SOARS, StarLogo, Sugarscape, Swarm, UrbanSim, VisualBots, VSEit
Education/Teaching	AgentScript, AgentSheets, BehaviourComposer (2D/3D), ExtendSim (2D/3D), Framsticks (2D/3D), JAS-mine, MIMOSE, MOBIDYC, NetLogo (2D/3D), Scratch (2D/3D), SeSAm, SimSketch, Simul8, StarLogo, StarLogo TNG (3D), Sugarscape, VisualBots, VSEit
Cloud computing/Virtualised datacentres	CloudSim
Geographic Information System (GIS), Geographic Automata System (GAS)	Cormas, Envision, GAMA (2D/3D), Insight Maker, MATSim, OBEUS, Pandora, Repast-J/Repast-3, Repast HPC, Repast Simphony (2D/3D), SOARS, TerraME
Aviation, Flight or air-traffic control, Ground transportation/Mobility planning systems	CybelePro, D-OMAR, ExtendSim (2D/3D), FlexSim (2D/3D), MACSimJX, MATSim, Mobility testbed, SimEvents (MATLAB®), Simio (2D/3D), SimJr, Swarm, UrbanSim
Consumer products, Manufacturing, Production (factory based optimized plans for different requirements), Logistics/Distribution/Supply Chains (coordination, storage layout optimization)	AnyLogic (2D/3D), CRAFTY, ExtendSim (2D/3D), FlexSim (2D/3D), HLA_Agent, HLA_RePast, PDES-MAS, Repast-J/Repast-3, Repast HPC, Repast Simphony (2D/3D), SeSAm, SimEvents (MATLAB®), Simio (2D/3D), Simul8, Swarm
Urban Planning (accessibility studies with dynamic populations)	AnyLogic (2D/3D), CRAFTY, Envision, GAMA (2D/3D), JAS-mine, Modgen, OBEUS, UrbanSim
Microscopic pedestrian crowd or mapping passenger flow (market improvement & evacuation of buildings)	Brahms, HLA_Agent, HLA_RePast, PDES-MAS, PedSim, Repast-J/Repast-3, Repast HPC, Repast Simphony (2D/3D), SeSAm, Simio (2D/3D)
Political Phenomena	Ascape, PS-I, VisualBots
Military-combat/War-fighting/Air-defense Scenarios	ExtendSim (2D/3D), HLA_Agent, HLA_RePast, PDES-MAS, SEAS (2D/3D), SimEvents (MATLAB®), Simio (2D/3D), SimJr
Financial market's stocks/Securities, Macroeconomic activity	Altreva adaptive modeler, JAMEL, JASA
Large-scale parallel/Distributed computing clusters & high performance supercomputers	Agent Cell (2D/3D), Ecolab, FLAME, FLAME GPU (3D), GridABM, HLA_Agent, HLA_RePast, MATSIM, Pandora, PDES-MAS, Repast-J or Repast-3, Repast HPC, Repast Symphony (2D/3D), Swarm

issue de [Abar et al., 2017]

Panorama des outils

Outils orientés simulation multi-agent

- Outils spécialisés orientés simulation multi-agent
  - ATOM (Agent-Based Computational Economics), ARCHISIM, MATSim, MITSIMlab (Transport), ...
  + réutilisation de certaines parties du modèle
  - adaptation pour une nouvelle plateforme

- Outils génériques orientés simulation multi-agent
  - AnyLogic, GAMA, MASON, SWARM, CORMAS, TurtleKit, Repast, Netlogo, JAS-MINE, ...
  + utilisation du même outil pour différents modèles
  - construction d'un modèle opérationnel pour chaque simulation

- Outils multi-agents généraux
  - Jason, JADE/TAPAS, JADE/PlaSMA, MADKIT, ...
  + connaissance générale de l'outil
  - adaptation nécessaire pour la mise en place du modèle de simulation

adapté du cours de F. Balbo 2017

Repast (https://repast.github.io/index.html)

Outils orientés simulation multi-agent

- Famille de plateforme de simulations (Recursive Porous Agent Simulation Toolkit)
- Créée en 2000 / académique / Open Source / Gratuite
- Acteurs : University of Chicago et Argonne National Laboratory
- Versions avec différents langages : Python, C#, ReLogo, Java, C++ / exécutable sur différents environnements matériels (clusters)
- Différentes visualisations possibles (2D, 3D)
- Différents types d'espaces dans lesquels situer les agents
- Disponibilité d'outils d'apprentissages

Repast (suite)

Outils orientés simulation multi-agent

- Concepts : modèle, agents et espace qui définit l'environnement dans lequel les agents sont situés et exécutent leurs actions

```java
public class Zombie {

    private ContinuousSpace<Object> space;
    private Grid<Object> grid;
    private boolean moved;

    public Zombie(ContinuousSpace<Object> space, Grid<Object> grid) {
        this.space = space;
        this.grid = grid;
    }

    @ScheduledMethod(start = 1, interval = 1)
    public void step() {
        // get the grid location of this Zombie
        GridPoint pt = grid.getLocation(this);

        // use the GridCellNgh class to create GridCells for
        // the surrounding neighborhood.
        GridCellNgh<Human> nghCreator = new GridCellNgh<Human>(grid, pt,
                Human.class, 1, 1);
        List<GridCell<Human>> gridCells = nghCreator.getNeighborhood(true);
        SimUtilities.shuffle(gridCells, RandomHelper.getUniform());

        GridPoint pointWithMostHumans = null;
        int maxCount = -1;
        for (GridCell<Human> cell : gridCells) {
            if (cell.size() > maxCount) {
                pointWithMostHumans = cell.getPoint();
                maxCount = cell.size();
            }
        }
        moveTowards(pointWithMostHumans);
        infect();
    }

    public void moveTowards(GridPoint pt) {
        // only move if we are not already in this grid location
```

issue de doc. Repast

GAMA (http://gama-platform.org/ https://github.com/gama-platform)

Outils orientés simulation multi-agent

- Modélisation et exécution de simulations ayant une composante spatiale explicite (GIS Agent-based Modeling Architecture).
- Créée en 2010 / académique / Open Source / Gratuite
- Acteurs :
  - IRD/UPMC international research unit UMMISCO, MSI Research Team, Vietnam National University, Hanoi, Vietnam, UMR 6228 IDEES, CNRS/University of Rouen, France, UMR 5505 IRIT, CNRS/University of Toulouse 1, France, DREAM Research Team, University of Can Tho, Vietnam, UMR 8623 LRI, CNRS/University Paris-Sud, France
- Basé sur la plateforme Eclipse
- GAML : langage agent, de haut niveau, impératif s'appuyant sur java
- Instanciation possible des agents à partir d'ensembles de données, de SIG
- Utilisée dans des projets large échelle (plusieurs millions d'agents)

GAMA (suite) / GAML (Langage agent de GAMA)

Outils orientés simulation multi-agent

- concepts : species, attributes, actions, behaviors, skills, aspects
- “tout est agent”
- extensible : possibilité de plusieurs architectures d’agents (des plus simples à BDI, etc)
- Déclaration possible de plusieurs interfaces, de vues sur la simulation

```
model burst_the_baloon

global{
    float worldDimension <- 5#m;
    geometry shape <- square(worldDimension);
    int nbBaloonDead <- 0;

    reflex buildBaloon when:(flip(0.1)) {
        create species:balloon number:1;
    }

    reflex endSimulation when:nbBaloonDead>10 {
        do halt;
    }
}
```

issue de doc. GAMA

GAMA (suite) / GAML (Langage agent de GAMA)

Outils orientés simulation multi-agent

- concepts : species, attributes, actions, behaviors, skills, aspects
- “tout est agent”
- extensible : possibilité de plusieurs architectures d’agents (des plus simples à BDI, etc)
- Déclaration possible de plusieurs interfaces, de vues sur la simulation

```
model burst_the_baloon

global{
    float worldDimensio
    geometry shape <- s
    int nbBaloonDead <-

    reflex buildBaloon
        create species:
    }

    reflex endSimulatio
        do halt;
    }
}
```

```
species balloon {
    float balloon_size;
    rgb balloon_color;
    init {
        balloon_size <- 0.1;
        balloon_color <- rgb(rnd(255),rnd(255),rnd(255));
    }

    reflex balloon_grow {
        balloon_size <- balloon_size + 0.01;
        if (balloon_size > 0.5) {
            if (flip(0.2)) {
                do balloon_burst;
            }
        }
    }

    float balloon_volume (float diameter) {
        float exact_value <- 2/3*#pi*diameter^3;
        float round_value <- (round(exact_value*1000))/1000;
        return round_value;
    }

    action balloon_burst {
        write "the baloon is dead !";
        nbBaloonDead <- nbBaloonDead + 1;
        do die;
    }

    aspect balloon_aspect {
        draw circle(balloon_size) color:balloon_color;
        draw text:string(balloon_volume(balloon_size)) color:#black;
    }
}
```

issue de doc. GAMA

GAMA (suite) / GAML (Langage agent de GAMA)

Outils orientés simulation multi-agent

- concepts : species, attributes, actions, behaviors, skills, aspects
- “tout est agent”
- extensible : possibilité de plusieurs architectures d’agents (des plus simples à BDI, etc)
- Déclaration possible de plusieurs interfaces, de vues sur la simulation

```
model burst_the_baloon

global{
    float worldDimensio
    geometry shape <- s
    int nbBaloonDead <-

    reflex buildBaloon
        create species:
    }

    reflex endSimulatio
        do halt;
    }
}
```

```
species balloon {
    float balloon_size;
    rgb balloon_color;
    init {
        balloon_size <- 0.1;
        balloon_color <- rgb(rnd(255),rnd(255),rnd(255));
    }

    reflex balloon_grow {
        balloon_size <- balloon_size + 0.01;
        if (balloon_size > 0.5) {
            if (flip(0.2)) {
                do balloon_burst;
            }
        }
    }

    float balloon_volume (float diameter) {
        float exact_value <- 2/3*#pi*diameter^3;
        float round_value <- (round(exact_value*1000))/1000;
        return round_value;
    }

        draw text:string(balloon_volume(balloon_size)) color:#black;
    }
}
```

```
experiment my_experiment type:gui
{
    output{
        display myDisplay {
            species balloon aspect:balloon_aspect;
        }
    }
}
```

issue de doc. GAMA

Plan

Champs d'action

Outils multi-agents généraux

- Propriétés des applications cibles :
  - Absence de vision monolithique
  - Multi-* (sites, expertise, domaines, points de vue, décisions, buts, motivations, ...)
  - Développement incrémental et collaboratif
  - Exécution continue, Adaptation
  - De plus en plus centrées Utilisateur

- Caractéristiques intrinsèques :
  - Distribution logique et le plus souvent physique
  - Absence de contrôle central
  - Couplage faible entre entités et applications du système
  - Hétérogénéité
  - Traitement intensif de connaissances
  - Ouverture : perméabilité, évolution en taille et structure
  - Délégation partielle ou totale des décisions des utilisateurs au système

Panorama / Types d'outils

Outils multi-agents généraux

- Plateformes (supports à l'exécution du système)
  - Plateformes agents : support à l'exécution d'agents et à leurs interactions
  - Plateformes multi-agents : support à l'exécution d'agents, d'environnements, d'organisations
- Langages de programmation (approches déclaratives, impératives ou hybrides)
- Frameworks fondés sur un ou plusieurs langages de programmation et leurs plateformes

Panorama / Caractéristiques

Outils multi-agents généraux

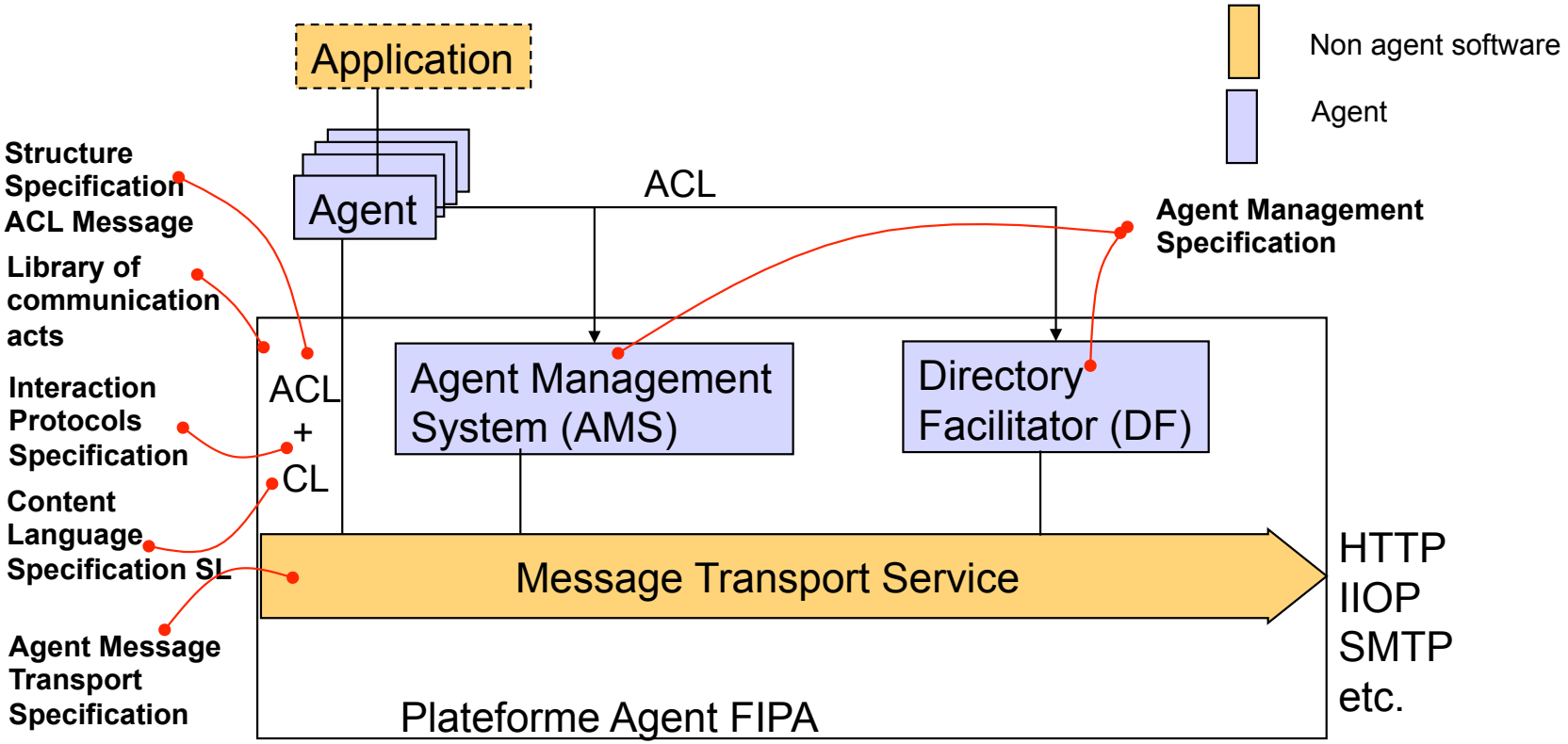
- Utilisable dans un objectif d'intégration multi-agent avec éventuellement possibilité d'utilisation dans un objectif de simulation multi-agent

- Génériques ou dédiées à un domaine

- Mise en oeuvre ou non de principes de :
  - Modularité, encapsulation
  - Séparation des préoccupations de la conception à la programmation
  - Réutilisabilité, support pour l'intégration d'applications existantes

Plateforme FIPA (http://www.fipa.org/)

Outils multi-agents généraux – Plateformes agent / multi-agent



ACL = Agent Communication Language

JADE (http://jade.tilab.com/)

- La plus utilisée
  - pour le développement d'applications
  - pour le support d'exécution d'autres outils
- Conforme aux spécifications de la FIPA, basée sur Java
- De nombreuses extensions dont :
  - WADE : outil de workflow multi-agent
  - AMUSE : environnement social multi-utilisateur basé sur approche agent

Autres plateformes:

- Java-based Intelligent Agent Componentware (JIACv) (open source, java, pour le développement d'applications large échelle, distribuées)

Jason (http://jason.sourceforge.net/)

- Langage de programmation d'agent hybride, open source, version étendue de AgentSpeak(L)
- Créé en 2003 / académique / Open Source / Gratuit
- Acteurs : PUCRS et UFSC (Brésil)
- Basé sur le modèle BDI, cycle de déliberation BDI
- intégration de code existant par le moyen d'actions internes
- de nombreuses extensions disponibles (e.g. Argo for Jason pour la programmation de robots, Javino, bibliothèqe pour la communication entre Jason et Raspberry+Arduino)
- Langage de programmation fortement utilisé dans le domaine

Jason (suite)

Outils multi-agents généraux – Langages de programmation agent

- concepts du langage : beliefs, goals, plans, actions
- communication inter-agent basée sur les actes de langage (belief, goals, plans avec annotation des sources d'information)
- possibilité de redéfinir les étapes du cycle de déliberation (ajout de fonction de trust, modification de révision de croyance, de perception, de communication, d'action, de sélection des intentions)

Example: bob.asl

```
happy(bob).
!say(hello).
+!say(X) : happy(bob)
<- .print(X).
```

Example: bob.asl

```
+happy(A) <- !say(hello(A)).
+!say(X) : not today(friday)
<- .print(X); !say(X).
+!say(X) : today(friday)
<- .print(stop).
-happy(A) : .my_name(A)
<- .drop_intention(say(_)).
```

Jason (suite)

Outils multi-agents généraux – Langages de programmation agent

- concepts du langage : beliefs, goals, plans, actions
- communication inter-agent basée sur les actes de langage (belief, goals, plans avec annotation des sources d'information)
- possibilité de redéfinir les étapes du cycle de déliberation (ajout de fonction de trust, modification de révision de croyance, de perception, de communication, d'action, de sélection des intentions)

Example: alice.asl

```
!start.
+!start <-.send(bob,tell,happy(bob));
          .send(bob,tell,happy(alice));
          .send(bob,achieve,count).
```

Example: bob.asl

```
happy(bob).
!say(hello).
+!say(X) : happy(bob) <- .print(X).
+!count : count(0) <- inc.
```

Autres

- Commerciaux :
  - JACK Agent Language (JAL) sous-ensemble de la plateforme JACK, développé et distribué par Agent Decision-Making Software (AOS), basé sur Java
  - ActiveEdge basé sur java, sur l'architecture d'agent cognitif Cougaar , développée et distribuée par COUGAAR Software Inc. (CSI)
- Gratuites et open source :
  - Jadex, plateforme agent open source s'appuyant sur l'approche par composants actifs, définition de processus dirigés par des buts, de workflows (développé par Univ. Hambourg)
  - BDI4Jade au dessus de JADE, développée par FURG Brazil.
  - GOAL développé par Univ. Tech. Delft (NL)

Autres langages de programmation

Outils multi-agents généraux

Plateformes et langages de programmation d'organisation :

- MOISE, framework open-source, développé par UFSC et USP Brésil, Mines Saint-Etienne, France
- Operetta framework open-source, developpé par Université de Technologie de Delft (NL), Université Politechnique de Catalogne (SP), Université d'Aberdeen (UK)

Plateformes et Langages de programmation d'environnement

- CArtAgO framework open-source pour la programmation et l'exécution d'environnements multi-agents, développé par l'Univ. Bologne Italie
- Environment Interface Standard (EIS), proposition d'interface "standard" en Java pour connecter des agents à des entités contrôlables dans l'environnement

SARL/JANUS (http://www.sarl.io/ http://www.janusproject.io/)

Outils multi-agents généraux – Frameworks de programmation multi-agent

- SARL fournit un ensemble de concepts pour le développement de systèmes holoniques (agents composés d'autres agents avec éventuellement de nouvelles capacités)
- basé sur Java
- Créée en 2008 (Janus), 2014 (SARL) / Académique / Open Source / Gratuite
- Acteurs : UTBM France, Tucuman Univ. Argentine
- Janus plateforme multi-agent fournit le support d'exécution pour un déploiement d'applications SARL dans un environnement web, entreprise et sur desktop

SARL/JANUS (suite)

Outils multi-agents généraux – Frameworks de programmation multi-agent

- Concepts principaux : événement, agent (attributs, contextes, comportements), capacité (ensemble d'actions), compétence (implémentation d'une capacité), espace et adresse

```
agent PingAgent {
    uses DefaultContextInteractions
    on Pong {
        emit(new Ping( occurrence.index + 1 ))
            [ it == occurrence.source ]
    }
    on Initialize {
        emit( new Ping(0) )
    }
}
```

SARL/JANUS (suite)

- Concepts principaux : événement, agent (attributs, contextes, comportements), capacité (ensemble d'actions), compétence (implémentation d'une capacité), espace et adresse

```
agent PingAgent {
    uses DefaultContextInteractions
    on Pong {
        emit(new Ping( occurrence.index + 1 ))
    agent PingAgent {
        uses DefaultContextInteractions, Schedules
        on Pong {
            emit(new Ping( occurrence.index + 1 ))
                [ it == occurrence.source ]
}       }
        on Initialize {
            val task = task("waiting_for_partner")
            task.every(1000) [
                if (defaultSpace.participants.size > 1) {
                    emit( new Ping(0) )
                    task.cancel
                }
            ]
        }
    }
```

issue de doc. SARL

JaCaMo (http://jacamo.sourceforge.net https://github.com/jacamo-lang/jacamo)

- JaCaMo est un framework de programmation orienté multi-agent basé sur l'intégration de Jason (Agents), CArtAgO (Environnement) et MOISE (Organisation)
- Créé en 2009 / académique / Open Source / Gratuite
- Acteurs : PUCRS et FUSC (Brésil), Univ. Bologne (Italie), Mines Saint-Etienne (France)
- Intégration avec d'autres technologies multi-agents ou issues d'autres domaines

JaCaMo (suite)

Outils multi-agents généraux – Frameworks de programmation multi-agent

Example: alice.asl

```
!start.
+!start <-.send(bob,tell,happy(bob));
          .send(bob,tell,happy(alice));
          .send(bob,achieve,count).
```

Example: bob.asl

```
happy(bob).
!say(hello).
+!say(X) : happy(bob) <- .print(X).
+!count : count(0) <- inc.
```

JaCaMo (suite)

Outils multi-agents généraux – Frameworks de programmation multi-agent

Example: alice.asl

```
!start.
+!start <-.send(bob,tell,happy(bob));
          .send(bob,tell,happy(alice));
          .send(bob,achieve,count).
```

Example: bob.asl

```
happy(bob).
!say(hello).
+!say(X) : happy(bob) <- .print(X).
+!count : count(0) <- inc.
```

```
public class Counter extends Artifact {
    void init(int initialValue) {
        defineObsProperty("count", initialValue);
    }

    @OPERATION void inc() {
        ObsProperty prop = getObsProperty("count");
        prop.updateValue(prop.intValue()+1);
    }
}
```

JaCaMo (suite)

Outils multi-agents généraux – Frameworks de programmation multi-agent

Example: alice.asl

```
!start.
+!start <-.send(bob,tell,happy(bob));
          .send(bob,tell,happy(alice));
          .send(bob,achieve,count).
```

Example: bob.asl

```
happy(bob).
!say(hello).
+!say(X) : happy(bob) <- .print(X).
+!count : count(0) <- inc.
```

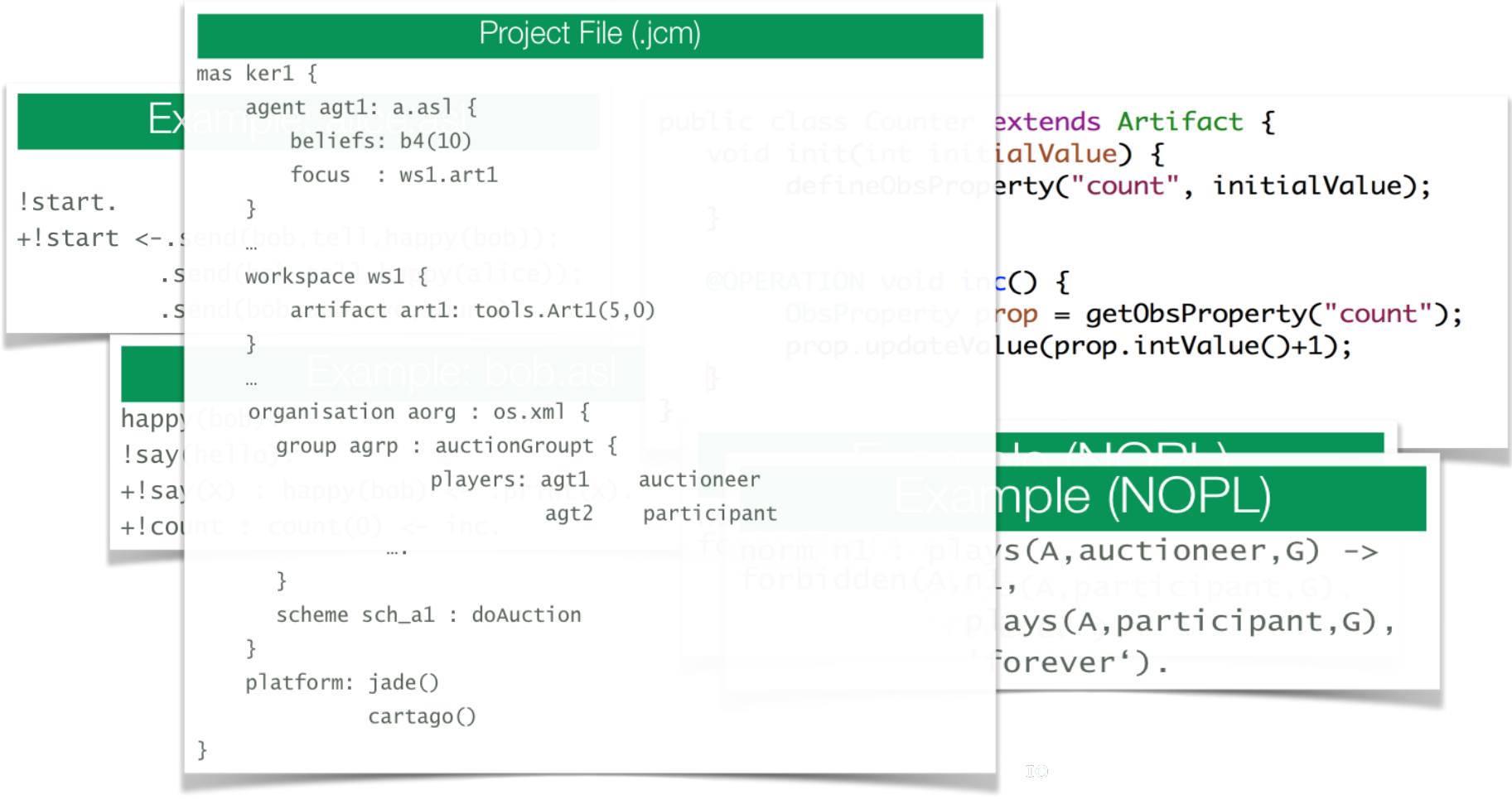
```
public class Counter extends Artifact {
    void init(int initialValue) {
        defineObsProperty("count", initialValue);
    }

    @OPERATION void inc() {
        ObsProperty prop = getObsProperty("count");
        prop.updateValue(prop.intValue()+1);
    }
}
```

Example (NOPL)

```
norm n1 : plays(A,auctioneer,G) ->
forbidden(A,n1,
          plays(A,participant,G),
          'forever').
```

JaCaMo (suite)

Outils multi-agents généraux – Frameworks de programmation multi-agent



Autres (Commerciales)

- BRAHMS developpée par EJENTA pour le développement d'application pour la NASA, pour l'assistance à diabétiques, etc (développement initialement conduit par Intelligent Systems Division at NASA Ames Research Center)
- Living Systems Process Suite (LSPS) développée par Whitestein technologies pour la gestion de processus métiers dirigés par les buts

Autres (Gratuites et open source)

- Siebog basé sur Jason, développée par Université de Novi Sad, Serbie
- ENMASSE basée sur Javascript, s'exécute sur node.js et dans le navigateur, utililisée pour la simulation et l'intégration, developpée par Almende
- Madkit basée sur java, le meta model AGRE/MASK, utilisée pour la simulation et l'intégration, développée par l'Univ. Montpellier
- ANTE basée sur java, dédiée à des systèmes mettant en oeuvre négociation, normes et confiance, développée par Université de Porto, Portugal
- Electronic Institutions - EI/EIDE basée sur java, dédiée à des institutions électroniques, développée par IIIA Barcelone, Espagne

Se reporter à [Aldewereld et al., 2016] pour plus de détails sur certaines de ces plateformes

Plan

Un état des lieux contrasté

Synthèse

- Domaines actifs avec productions de plateformes, langages :
  - avec différents status : libres ou non, commerciaux ou non, génériques ou dédiées
  - utilisés pour l'enseignement,
  - utilisés pour le développement d'applications de simulation, de résolution de problèmes, pour l'intégration et développement de systèmes socio-techniques

2005 AgentLink (Software Products for MAS, AgentLink, June 2002) et Agent Software Directory (état des lieux)

- 128 outils listés (en 2017, 53 toujours actifs)

2015 ICT COST Action IC1404 (état des lieux)

- Outils généraux (13 actifs sur 22 listés), Outils de simulation (4 actifs sur 6 listés), Outils dédiés (7 actifs sur 8 listés), Outils arrêtés (39), Méthodologies (14)

2017 Outils orientés simulation multi-agent [Abar et al., 2017] (état des lieux) : 85 recensés

Synthèse

:-) Passage d'une vision agent à une vision multi-agent avec la proposition d'outils matures !!!

:-(Absence de consensus sur les concepts et modèles

- $\rightsquigarrow$ utilisation de concepts et langages particuliers à chaque outil
- $\rightsquigarrow$ existence de plusieurs frameworks avec des environnements d'exécution particuliers (plateforme agent ou plateforme multi-agent)

- La plateforme d'exécution n'est plus au centre des préoccupations:
  - Respect des standard FIPA n'est plus un sujet de discussion
  - Jade est souvent utilisée comme couche support à l'exécution du système multi-agent
- Les modèles et concepts utilisés (des agents, aux environnements, aux organisations) sont au coeur des préoccupations
- Existence de compétitions pour comparer les outils :
  - Multiagent Programming Contest,
  - RoboCup,
  - ...

Quelques Perspectives

- Poursuivre l'étude des questions liées au debug, aux performances
- En simulation multi-agent, poursuivre les travaux sur les questions de distribution physique des simulations, de couplage de modèles
- Déboucher sur des consensus sur les concepts, sur des standards allant au delà des standards FIPA
- Etablir des liens avec d'autres domaines,
- Intégrer d'autres technologies complémentaires (Web sémantique / Autres plateformes matérielles, IoT, Web, Réalité augmentée, etc)

Bibliography I

Abar, S., Theodoropoulos, G. K., Lemarinier, P., and O'Hare, G. M. (2017).

Agent based modelling and simulation tools: A review of the state-of-art software.

Computer Science Review.

Aldewereld, H., Boissier, O., Dignum, V., Noriega, P., and Padget, J., editors (2016).

Social Coordination Frameworks for Social Technical Systems, volume 30 of *Law, Governance and Technology Series*.

Springer International Publishing, 1 edition.

Bergenti, F., Gleizes, M.-P., and Zambonelli, F. (2004).

Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook, volume 11.

Springer Science & Business Media.

Bibliography II

Ciortea, A., Boissier, O., Zimmermann, A., and Florea, A. M. (2017).

Give agents some REST: A resource-oriented abstraction layer for internet-scale agent environments.

In Larson, K., Winikoff, M., Das, S., and Durfee, E. H., editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 1502–1504. ACM.

Demazeau, Y. (1995).

From interactions to collective behaviour in agent-based systems.

In *Proc. of the 1st European Conf. on Cognitive Science. Saint-Malo*, pages 117–132.

Drogoul, A. (1993).

De la simulation multi-agents à la résolution collective de problèmes.

PhD thesis, Thesis at University of Paris IV.

Ferrand, N. (1997).

Modèles multi-agents pour l'aide à la décision et la négociation en aménagement du territoire.

PhD thesis, Université Joseph-Fourier-Grenoble I.