SARL: www.sarl.io
Agent-Oriented Programming Language

Sebastian Rodriguez  Nicolas Gaud  Stéphane Galland
Seminar at ISCOD / LSTI - Institut Henri Fayol – Saint Étienne
February 19th, 2014
Janus Experience

- Janus code base is almost 10 years old.
- Learned a lot of the DO’s and DON’Ts.
- API became complex and difficult to maintain.
- Constant need to refactor to include new features.
- New patterns have changed software development (IoC, Event-Driven Communication, Distributed Objects, etc.).
Expectations for SARL

- Nowadays we implement Agent with Object-Oriented concepts.
- Object-Oriented design with Object-Oriented concepts (interfaces, classes, methods, etc).
- Agents should be simple to extend.
- Provide the community a common discussion forum.
Design Principles

- Clear separation between Language and Platform related aspects
- Everything is distributed and it should be transparent.
- Massively parallel.
- Event-driven interactions.
- Platform- and architecture-independent.
- Coding should be fun (Ruby/Scala-like).
- All agents are holonic.
- There is not only one way of interacting but infinite.
Outline

1. Motivation
2. Design Principles
3. Main Concepts
   - Definitions
   - Built-in Capacities
4. Show me the Code!
5. Summary
6. Future works
An agent is an autonomous entity having some intrinsic skills to implement the capacities it exhibits.

An agent defines a Context.

An agent initially owns native capacities called Built-in Capacities.
Capacities and Skill

Capacity

Action
- A specification of a transformation of a part of the designed system or its environment.
- Guarantees resulting properties if the system before the transformation satisfies a set of constraints.
- Defined in terms of pre- and post-conditions.

Skill
A possible implementation of a capacity fulfilling all the constraints of its specification.
Context and Interactions

**Context**
- Defines the boundary of a sub-system.
- Collection of Spaces.
- Every Context has a Default Space.
- Every Agent has a Default Context, the context where it was spawned.

**Space**
Support of interaction between agents respecting the rules defined in various Space Specifications.
Space Specification

- Defines the rules (including action and perception) for interacting within a given set of Spaces respecting this specification.
- Defines the way agents are addressed and perceived by other agents in the same space.
- A way for implementing new interaction means.

Use case for organizational approach

- **Space Specification**: Organization
- **Space**: Group
- **Agent interface**: Behavior, Role
- **Addressing**: Role Address
Default Space: an Event Space

- Event-driven interaction space.
- Default Space of a context, contains all agents of the considered context.
- Event: the specification of some occurrence in a Space that may potentially trigger effects by a participant.
All agents have at least one External Context (the default one).

All agents participate in the Default Space of all Contexts they belong to.

The Janus Context is omnipresent.
1 Motivation

2 Design Principles

3 Main Concepts
   ■ Definitions
   ■ Built-in Capacities

4 Show me the Code!

5 Summary

6 Future works
A SARL Agent has inherently a set of **Built-in Capacities**

### Current Built-in Capacities

- ExternalContextAccess
- InnerContextAccess
- Behaviors
- Lifecycle
- Schedules
- DefaultContextInteractions
Behaviors Built-in Capacity

Behavior

Defines the actions to be performed on a given perception (Events) in a Space.

Agent

Inner Context

Skill Container

Behaviors Built-in Capacity

Behavior 1

Behavior 2

Behavior 3

Behavior n

Capacity 1

Capacity 2

Capacity 3

Capacity n

Space 1

Space 2

Default Space

Addr1

Addr2
Show me the Code!

Demo 😊
What does SARL currently provide?

- Defines general agent-technology concepts: Context, Space, Agent, Capacity, Skill.
- All agents are holons.
- Intuitive Syntax (and the associated Eclipse-based IDE).
- Extensible (Capacities and Skill)
- No single way of interacting imposed.
- Janus as SARL platform
  - Fully distributed.
  - Dynamic discovery of Kernels.
  - Automatic synchronization of kernels’ data (easy recovery).
  - Micro-Kernel implementation.
Future works

- Define the organizational extensions firstly based on CRIO then MOISE ☺: static and dynamic (normative).
- Define the concept of Environment.
  - Environment dynamics and interface.
  - How to define “objects” inside the environment. (artifacts?)
  - Simulation extension.
- Enforcing Pre- and post-conditions.
- Formal specification.
Thank you for your attention...