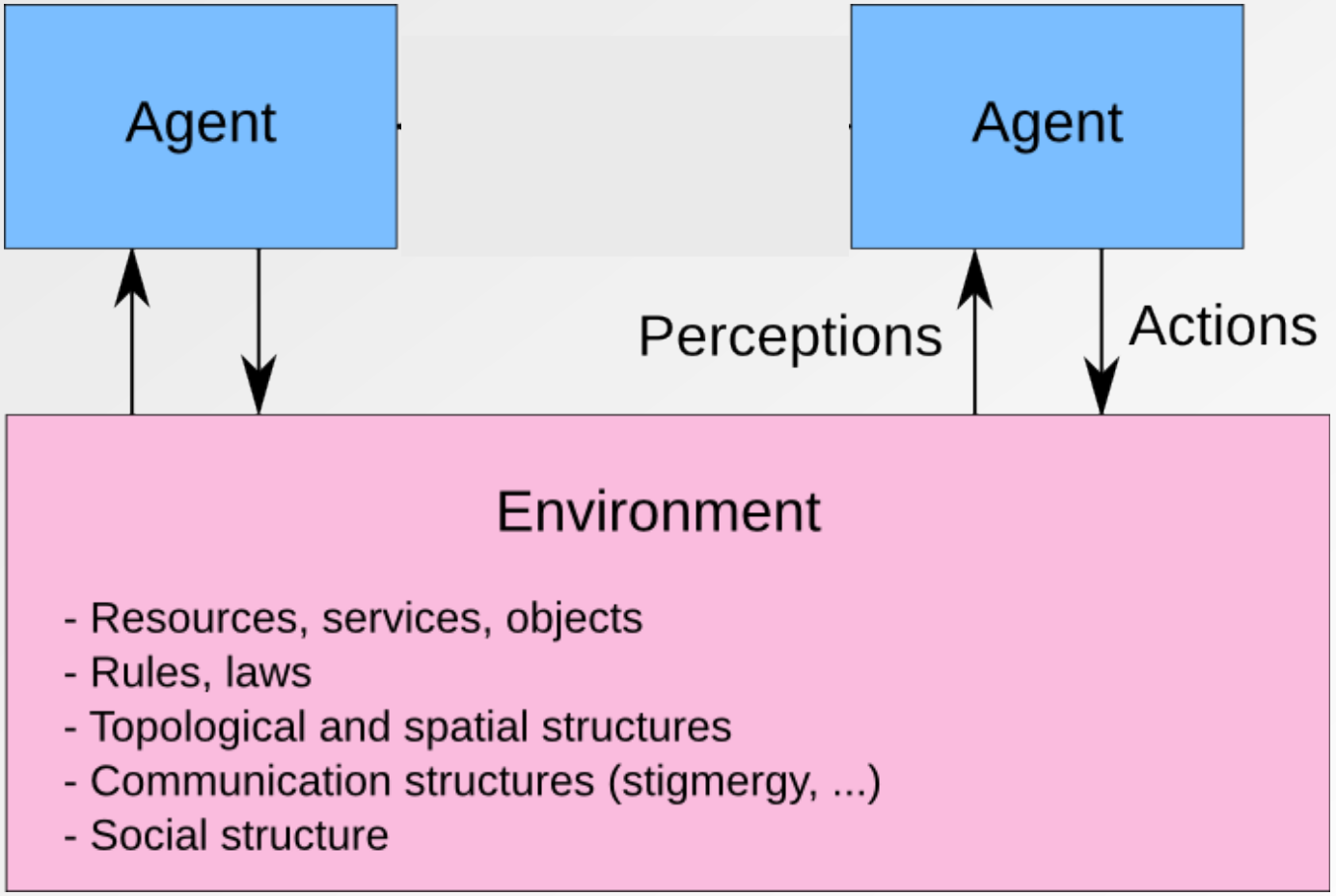
Contextualize Agent Interactions by Combining Communication and Physical Dimensions in the Environment

S. GALLAND, F. BALBO, N. GAUD,
S. RODRIGUEZ, G. PICARD, O. BOISSIER

13th International Conference on Practical
Applications of Agents and Multi-Agent Systems
PAAMS 2015

IRTES
sciences & ingénierie
Institut de Recherche sur les Transports, l'Énergie et la Société
utbm
université de technologie
Belfort-Montbéliard
UNIVERSIDAD
TECNOLOGICA
NACIONAL
Facultad Regional Tucumán
MINES
Saint-Étienne

Context: Agents & Environment

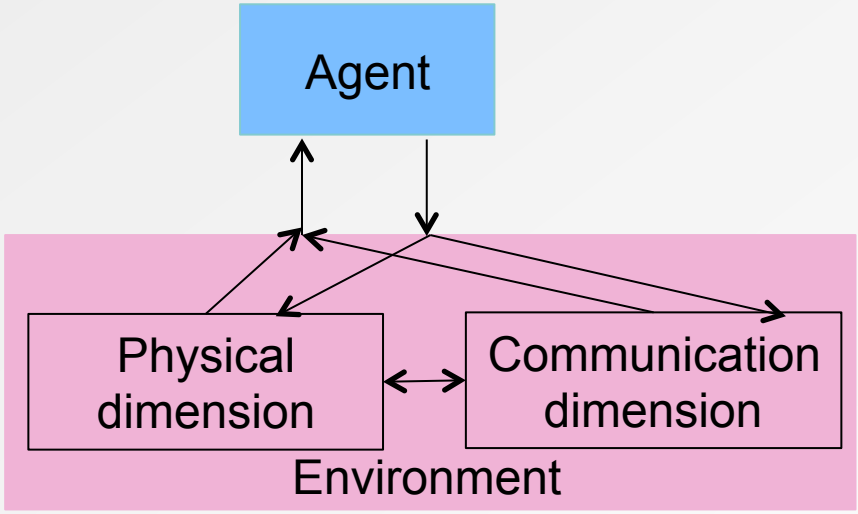


Environment has several dimensions [Odell, 2002]

- **Physical**
  - Principles and processes that govern and support a population of entities
  - Each agent has a body corresponding to its physical representation [Michel, 2004]
- **Communication**
  - Principles, processes and structures to transport information between agents
- **Social**
  - Principles, processes and structures to support coordinated interaction between agents in a communication environment

Problem

- **Hypothesis:** A change of state in a dimension can cause a change in another or several dimensions.
- **Solution 1:** agent as a propagation vector
- **Solution 2:** interactions between dimensions



How to model interactions between the dimensions?

Interactions among Dimensions outside of the agents

- **Multi-dimension diffusion / polymorphism**
  - An event/message can be simultaneously interpreted (differently) by several dimensions.

  e.g.: GPS Alert may change the social status and spatial indicator of dangerousness.
- **Propagation of interactions.**
  - An event/message in a given dimension generates another event/ message in the other dimension.

  e.g.: the detection of physical collision may trigger an emergency message in the communication dimension.
- **Constrained perception.**
  - A perception of an event/message in a dimension may be constrained by the properties associated with the other dimensions.

  e.g.: a traffic light (physical dimension) perceives only emergency vehicles (social status) that are close (physical dimension) to the light.

Approach

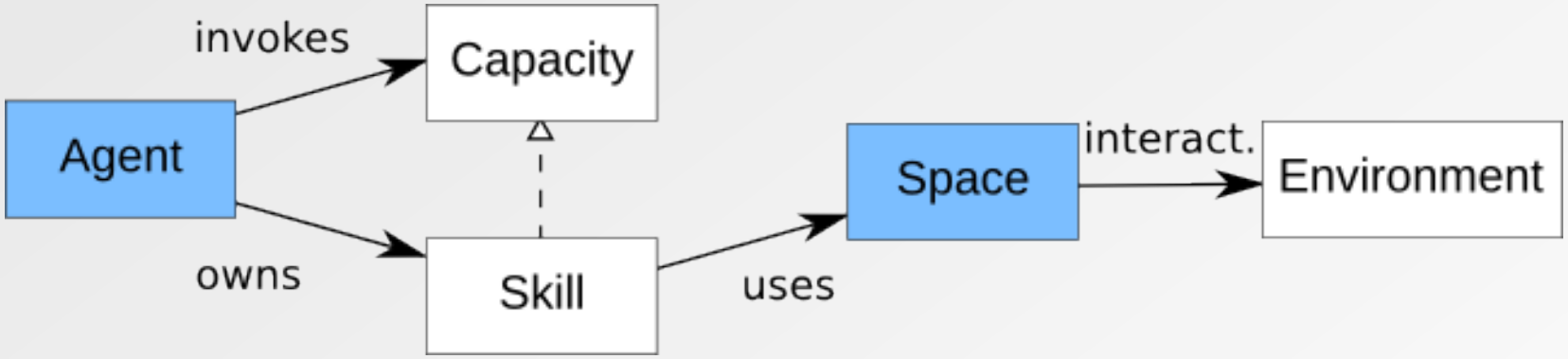
- **Modeling based on the SARL, general-purpose agent-oriented programming language.**

- **Major characteristics**
  - Agents are holonic (recursive agents)
  - Extendable interaction model (default is event-based)
  - Separation between the language and the running platform
  - Everything is performed in parallel without effort:
    - agent, behavior, etc.

www.sarl.io

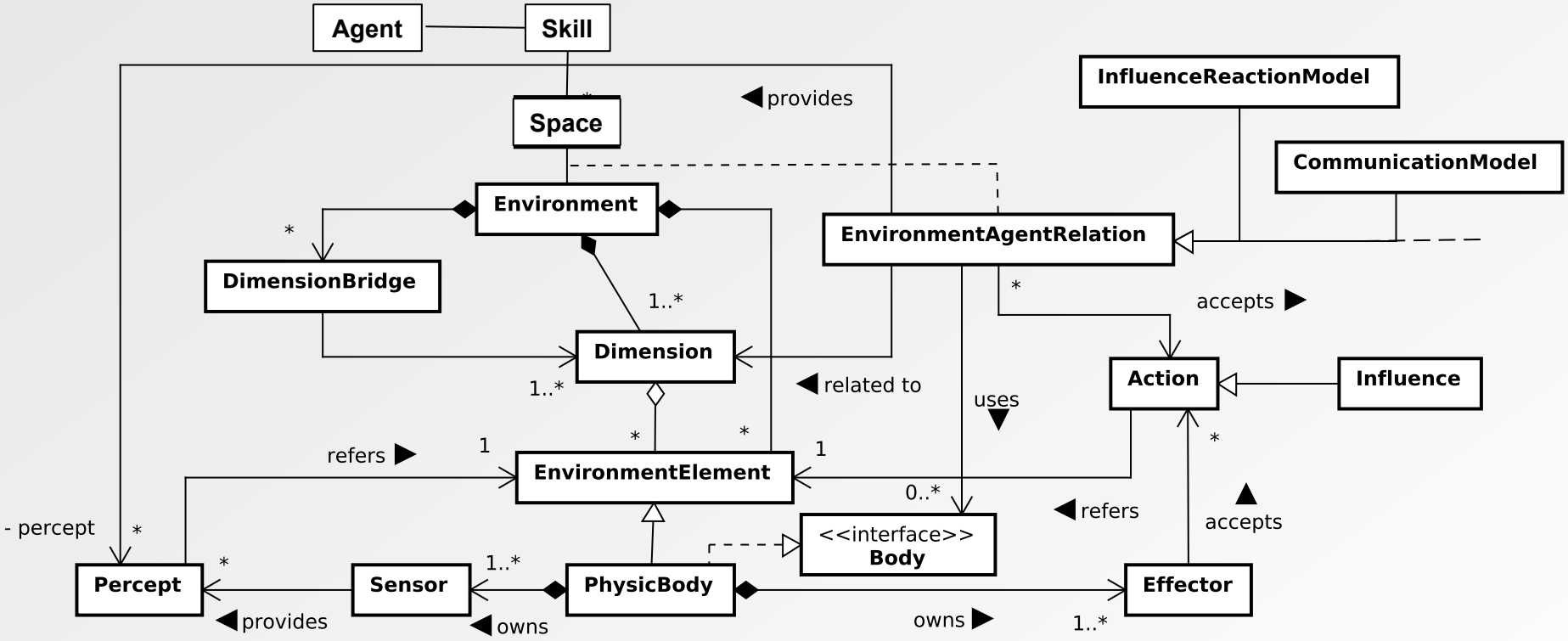
« SARL : a general-purpose agent-oriented programming language. »
Rodriguez S., Gaud N., Galland S. (2014). IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IEEE Computer Society Press, Warsaw, Poland.

SARL Main Concepts



- **Space:** interaction place between *agents* or between agents and their *environment* (e.g. EventSpace EventSpaceImpl)
- **Agent:** autonomous entity with a set of *skills* to realize the *capacities* it exhibits
- **Capacity:** specification of a collection of *actions*
- **Skill:** possible implementation of a *capacity*
  - Skills realized by behaviors mapping a collection of perceptions represented by Events to a sequence of Actions. Event is the specification of some occurrence in a Space that may potentially trigger effects by a listener

Proposal: Environment Model



Environment: Physical Dimension

- Class of systems (real or simulated) in which the agents have an explicit position, and in which the actions and perceptions are also located (situated MAS)

- Properties
  - Contains physical objects
  - Manages the bodies of agents
  - Enforces the laws of the Universe (e.g. the Laws of Physics)
  - Agents interact via a dedicated capacity
  - Agent-environment relationship model: Influence-reaction [Michel,2006]

Environment: Physical Dimension

- "PhysicalSpace" supporting the interaction between agents and the physical dimension
  - Emit influences from a specific body
  - Destroy the body related to an agent

```
space PhysicalSpace {
  def getBodyFactory: PhysicBodyFactory
  def putEnvironment(body: AgentBody,
                     perceptionListener: Agent)
  def influence(body : AgentBody,
                influences : Influence*)
  def destroyBody(body : AgentBody)
}

class PhysicalSpaceImpl extends EventSpaceImpl
                        implements PhysicalSpace {
  var env : Environment
  def influence(body : AgentBody,
                influences : Influence*) {
    for (i : influences) emit(i, env.scope)
  }
  ...
}
```

```
skill RoadEnvironmentSkill
          implements RoadEnvironmentCapacity {
  var body : AgentBody
  def install {
    body = bodyFactory.newInstance
    getSpace(PhysicSpace)
        .putEnvironment(body, owner)
  }
  def influence(inf: influence) {
    getSpace(PhysicSpace).influence(body, inf)
  }
  def uninstall {
    getSpace(PhysicSpace).destroyBody(body)
  }
  ...
}
```

Environment: Communication Dimension

- Exchanges of messages on the Internet

```
space InternetSpace {
  def emit(e : Message, scope : Scope)
  def register(agent : Agent) : Address
  def unregister(agentAddress : Address)
}

class InternetSpaceImpl extends EventSpaceImpl
                        implements InternetSpace {
  val env : Environment
  def emit(e : Message, scope : Scope) {
    e.destination = scope
    super.emit(e, new Scope(env))
  }
}
```

```
event Message {
  var destination : Scope
}

capacity InternetCapacity {
  def emit(e : Message,
           scope : Scope = null)
}

skill InternetSkill
      implements InternetCapacity {
  def install {
    getSpace(InternetSpace)
        .register(owner)
  }
  def emit(e : Message, scope : Scope=null) {
    getSpace(InternetSpace).emit(e, scope)
  }
  def uninstall {
    getSpace(InternetSpace).unregister (owner)
  }
}
```

Combining Dimensions

- **Content**
  - One instance of the models for each environment dimension
  - Rules of interaction between the dimensions, defined with:
    - a predicate p: rule activation condition
    - a function f: actions to perform when the rule is activated

- **Missions**
  - To compute the environment reactions from the agent influences
  - To compute the perceptions for each agent in the physical dimension
  - To propagate messages within the communication dimension

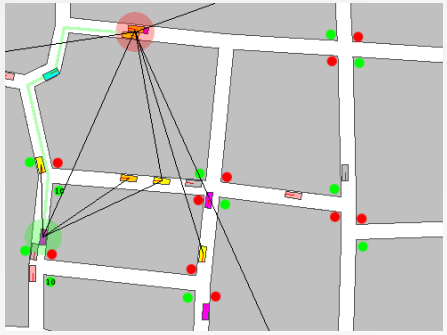
General Behavior of the Environment

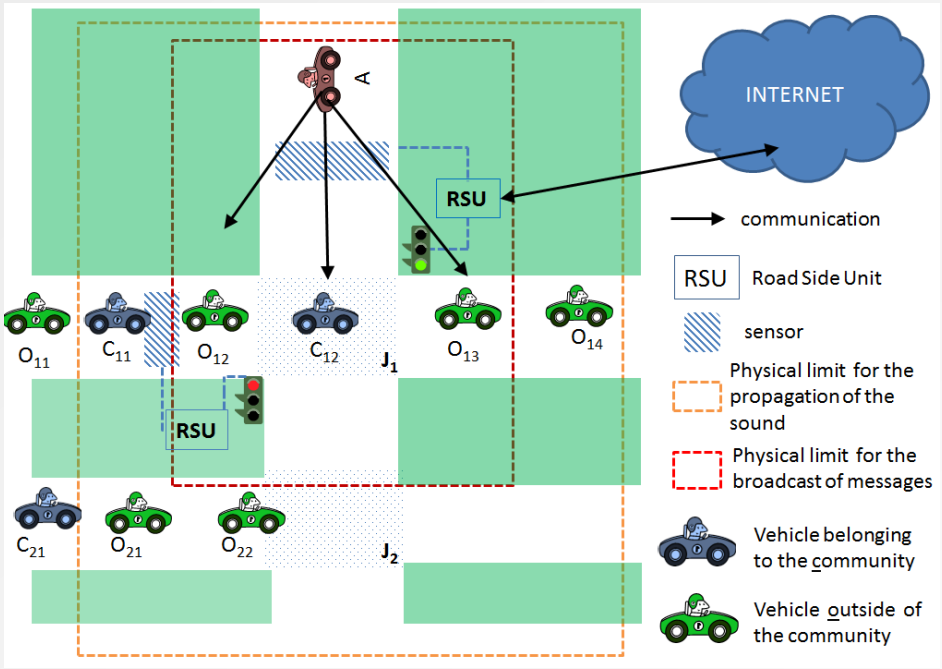
- When receiving an influence, the rules are applied, and the influence is preserved if no rule is deleting it
- When receiving a message, a similar algorithm is applied
- Influences and saved messages are stored for later use in the lifecycle

```
behavior Environment {
  var roads : RoadNetwork
  var physicSpace : space
  var communicationSpace: space
  ...
  on Influence {
    if (applyRules(occurrence, occurrence.object)) { saveInfluence(occurrence) }
  }
  on Message {
    for (participant : this.socialSpace.participants) {
      if (occurrence.scope.matches(participant) && applyRules(occurrence, participant))
      { saveMessage(occurrence) }
    }
  }
}
```

Proof of Concept: Traffic Simulation

- **Principles**
  - Simulation of traffic and car crashes
  - Simulation of the displacements of the emergency cars
  - "Green wave" for emergency cars
- **Dimensions of the environment**
  - Physical: Road network, traffic lights, road sensors
  - Communication: Wireless Network, RSU, Internet

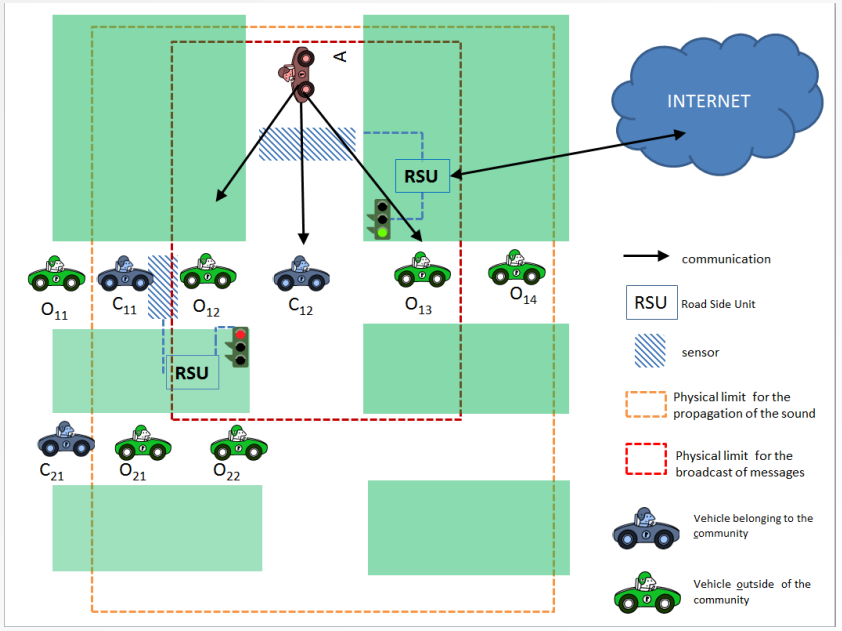




Rule examples (1/3): Interaction in one dimension constrained by the second

- **Agent A broadcasts a priority request within its community**
  - Priority request sent in the communication dimension, but the broadcast is limited according to the position of vehicles to the physical environment by the V2X propagation model

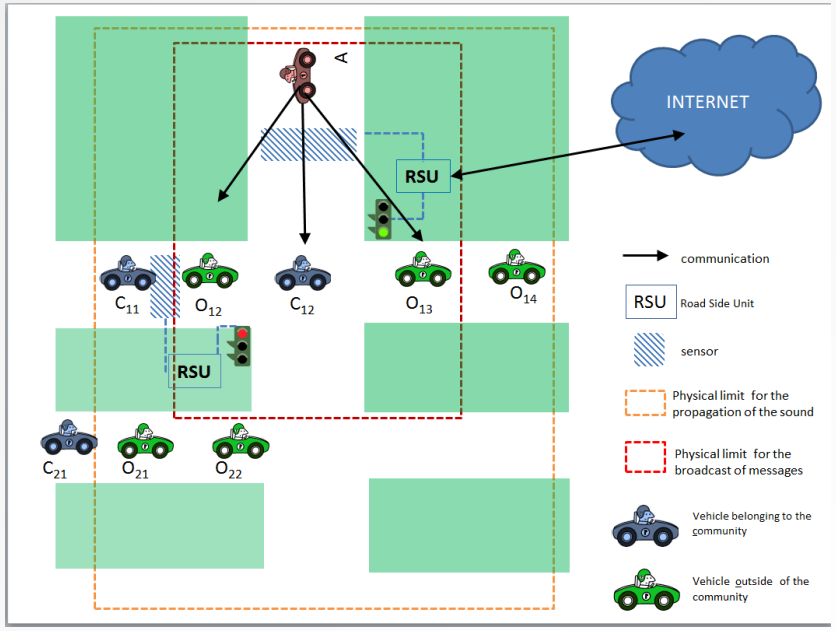
```
rules +=
    [ env, e, o |
      e instanceof PriorityRequestMessage ]
    =>
    [ env, e, o |
      e.scope =
        Scopes.addresses(
            env.roads.vehiclesAtDistance(
                  e.source, env.physicSpace.V2X_distance)
            )
    ]
```



Rule examples (2/3): *same interaction taking different forms in the two dimensions*

- **Agent A sends a priority request (resp. Siren influence) that is transformed into Siren influence (resp. priority request)**

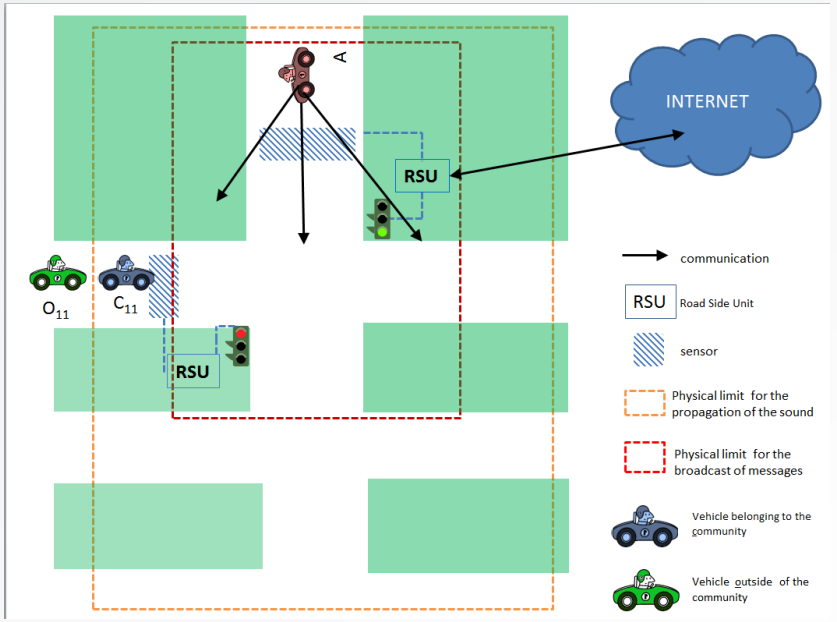
```
rules +=
    [ env, e, o |
       e instanceof Siren ]
    =>
    [ env, e, o |
       env.communicationSpace.emit(
           new PriorityRequestMessage(e.source)
       )
     ]

rules +=
      [ env, e, o |
         e instanceof PriorityRequestMessage ]
       =>
       [ env, e, o |
          env.physicSpace.influence(
             new Siren(e.source)
          )
        ]
```



Rule examples (3/3): *interaction initiated in a dimension generates an interaction in the other dimension*

- **A physical collision involves sending an alert message in the communication dimension**

```
rules +=
     [ env, e, o |
          e instanceof PhysicalCollision ]

          =>

     [ env, e, o |
          env.emit( new Alert(e.position))
     ]
```



Conclusions and Perspectives

- **Conclusions**
  - Definition of models of the physical and communication dimensions of the environment
  - Definition of relationships between two dimensions of the environment
  - Environment model combining the two dimensions
  - SARL concepts are suitable for modeling and implementation

- **Perspectives**
  - Refinement of the definition of the interaction between dimensions
  - Compare this approach with a JaCaMo one with Artifacts, Workspaces
  - Possible connections with GAMA, OpenTraffic, MATSIM, etc

Contextualize Agent Interactions by Combining Communication and Physical Dimensions in the Environment

S. GALLAND, F. BALBO, N. GAUD,
S. RODRIGUEZ, G. PICARD, O. BOISSIER

13th International Conference on Practical
Applications of Agents and Multi-Agent Systems
PAAMS 2015

IRTES
sciences & ingénierie
Institut de Recherche sur les Transports, l'Énergie et la Société
utbm
université de technologie
Belfort-Montbéliard
UNIVERSIDAD
TECNOLOGICA
NACIONAL
Facultad Regional Tucumán
MINES
Saint-Étienne

References

- Béhé F., Galland S., Gaud N., Nicolle C. and Koukam A. (2014). « An ontology-based metamodel for multiagent-based simulations. » International Journal on Simulation Modelling, Practice, and Theory, 40 :64–85.
- Cossentino M., Gaud N., Hilaire V., Galland S. and Koukam A. (2010). « ASPECS : an agent-oriented software process for engineering complex systems - how to design agent societies under a holonic perspective. » Autonomous Agents and Multi-Agent Systems, 2(2) :260–304.
- Michel F. (2004). « Formalism, tools and methodological elements for the modeling and simulation of multi-agents systems. » PhD thesis, LIRMM, Montpellier, France.
- Michel F. (2006). « Le modèle IRM4S : le principe influence/réaction pour la simulation de systèmes multi-agents. » In Journées Francophones sur les Systèmes Multi-Agents (JFSMA).
- Odell J. J., Van Dyke Parunak H., Fleischer M. and Brueckner S. (2002). « Modeling agents and their environment. » In Proceedings of the 3rd international conference on Agent-oriented software engineering III, AOSE'02, pages 16–31, Berlin, Heidelberg. Springer-Verlag.
- Rodriguez S., Gaud N., and Galland S. (2014). « SARL : a general-purpose agent-oriented programming language. » IAT 2014. Warsaw, Poland. IEEE Computer Society Press.

SARL: main Concepts (1/2)

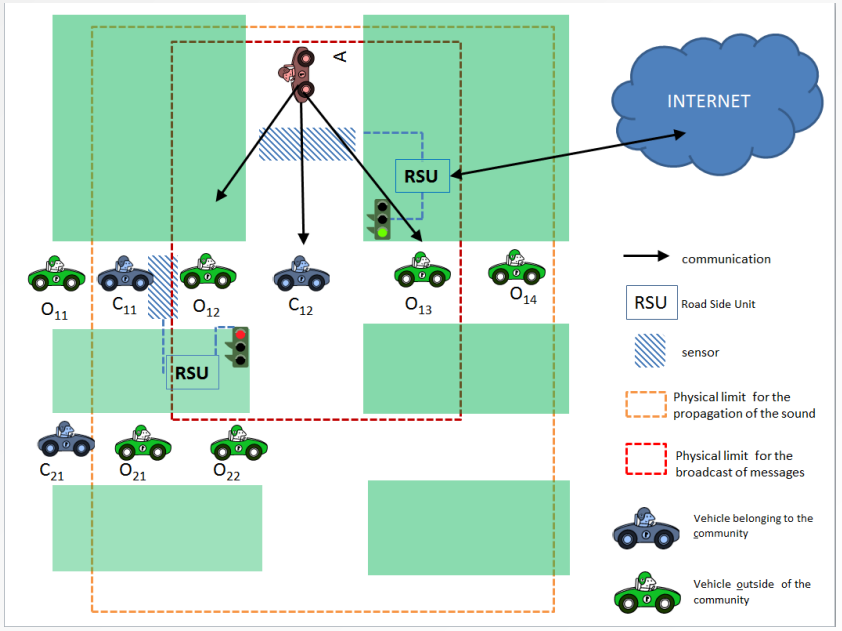
- **Multiagent System in SARL**
  - A set of agents interacting together in a set of distributed and shared spaces
- **Main concepts**
  - Agent, Capacity, Skill, Space
- **Main perspectives**
  - **Individual**: Behaviors (Agent, Capacity, Skill)
  - **Collective**: Interactions (Space, Event, etc.)
  - **Holonic**: Holons (Context)

« SARL : a general-purpose agent-oriented programming language. »
Rodriguez S., Gaud N., Galland S. (2014). IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IEEE Computer Society Press, Warsaw, Poland.

Examples of Rules (1/3)

- **Agent A priority request at traffic lights is accepted by the latter if the vehicle has priority status in the communication dimension**

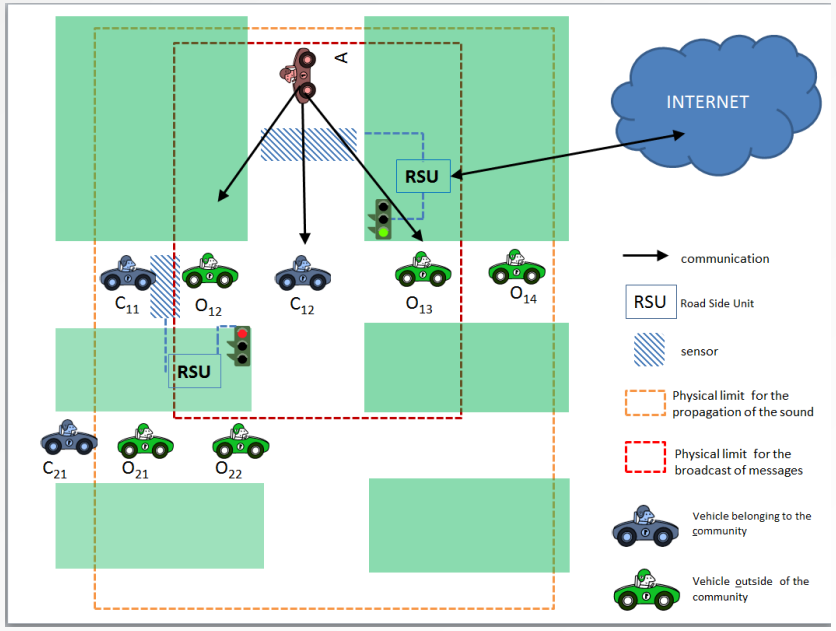
```
rules +=
        [ env, e, o |
          e instanceof PriorityRequest
          && o instanceof TraficLight ]

        ->

        [ env, e, o |
          env.communicationSpace
             .participant(e.source.ID).role
           instanceof PriorityVehicle ]
```



Examples of Rules (2/3)

- **A priority request to a traffic light increases the priorities of the road segments between the requester and the traffic light**

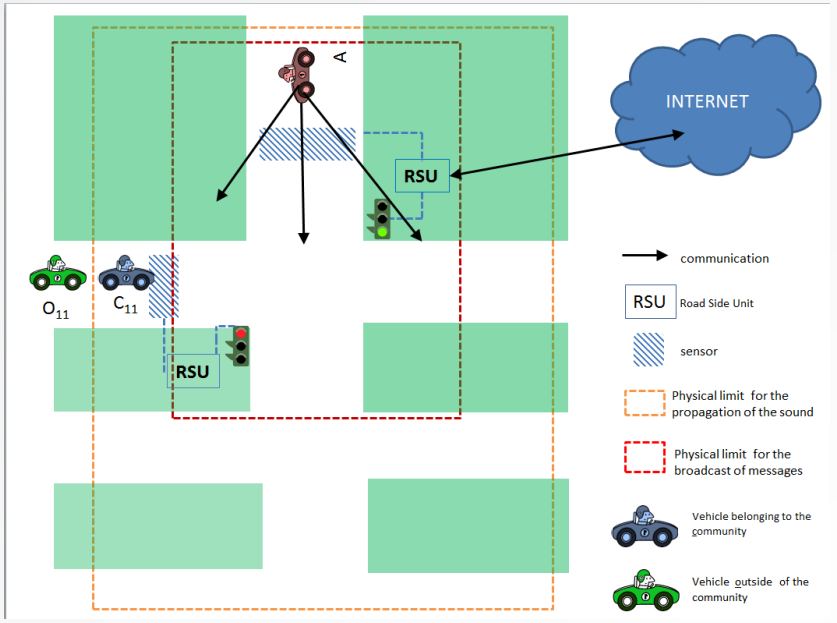
```
rules +=
        [ env, e, o |
          e instanceof PriorityRequest ]

        ->

        [ env, e, o |
          var sp = env.physicalSpace
          var p1 = sp.object(o.ID).position
          var p2 = sp.object(e.source.ID)
                     .position
          for (road : sp.roadsBetween(p1,p2)){
            road.priorityIndex += e.priority
          }
          return true ]
```



Examples of Rules (3/3)

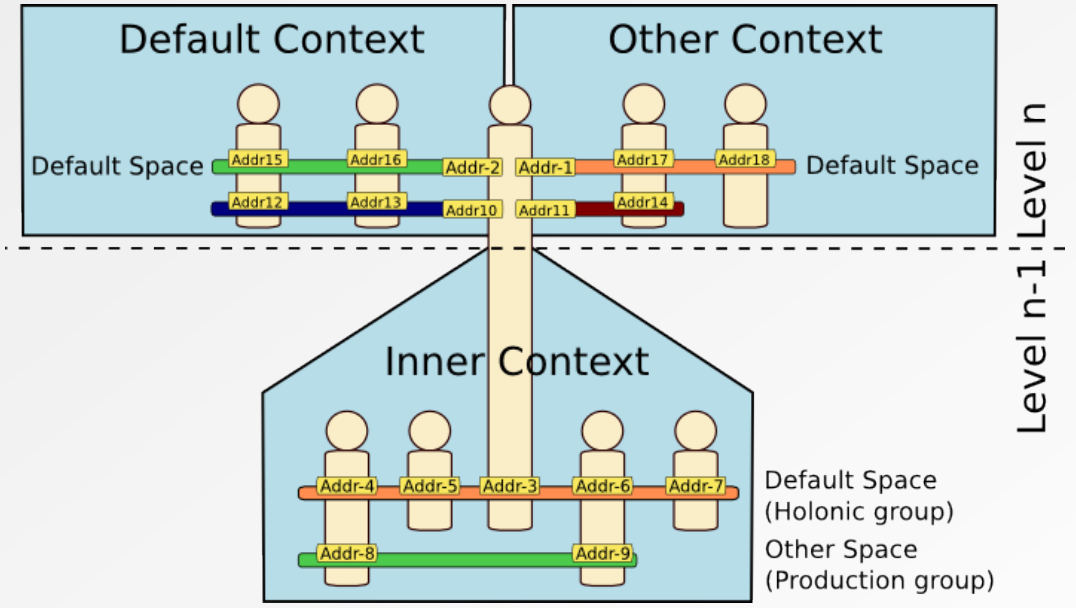
- **A physical collision involves sending an alert message in the communication dimension**

```
rules +=
          [ env, e, o |
            e instanceof PhysicalCollision ]

          ->

          [ env, e, o |
            env.emit( new Alert(e.position))
            return false ]
```



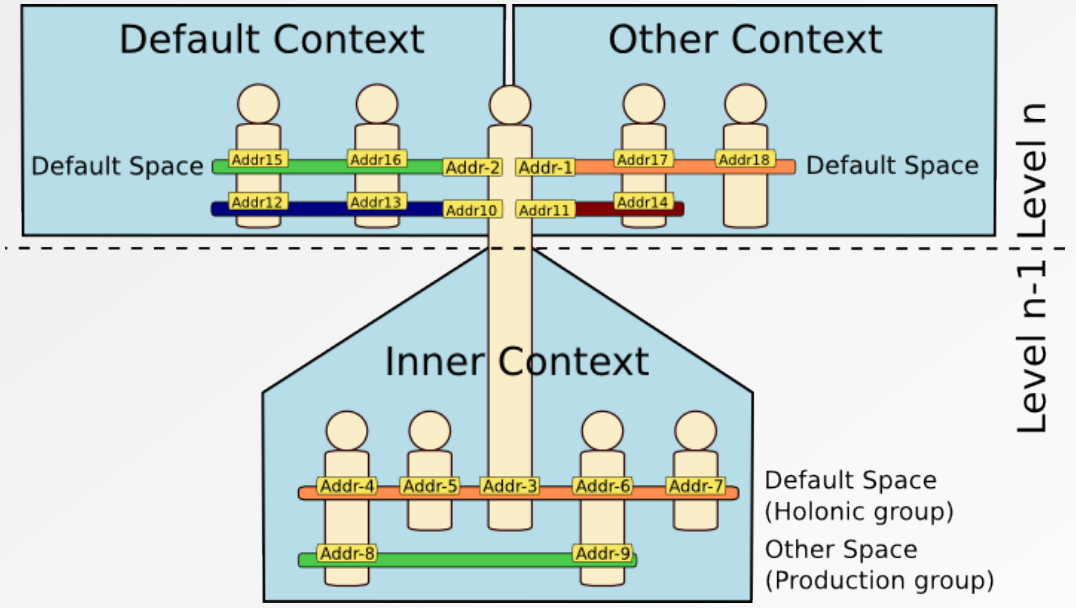
Collective Perspective on the Agent

- Member of its original/default context (where it has been spawn).
- Member of other contexts he joined.
- Interacts within the various spaces of its different contexts.



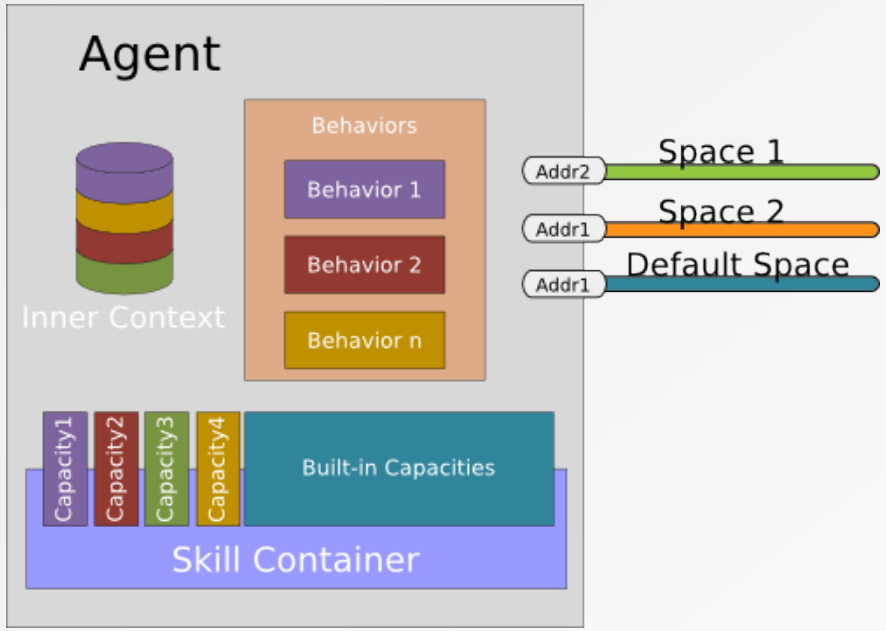
Holonic Perspective on the Agent

- Contains a context to which it belongs to:
  - the inner context
- Participates in the construction of hierarchies of agents.



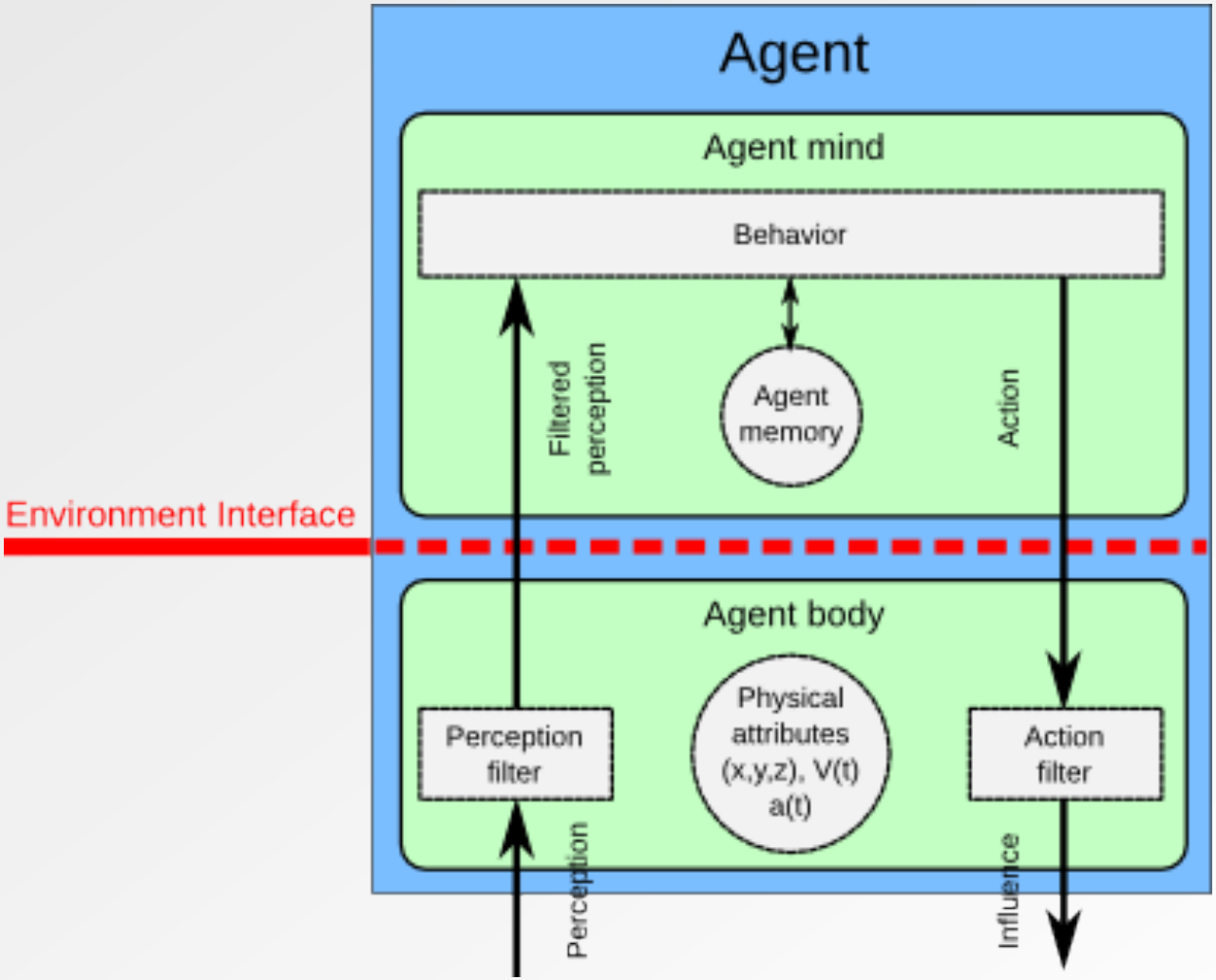
Agent and Capacity Concepts

- **Agent:**
  - An autonomous entity that has specific individual goals and the intrinsic ability to realize some capacities (thanks to its personal skills) [Cossentino, 2010].
  - Has intrinsic skills: communication, implementation of sub-behaviors, etc.
- **Capacity:**
  - Describes actions that could be performed by an agent.




```
agent HelloAgent {
  uses Lifecycle , Schedules
  on Initialize {
    println("Hello World")
    in (2000) [ killMe ]
  }
  on Destroy {
    println("Goodbye World !")
  }
}
```

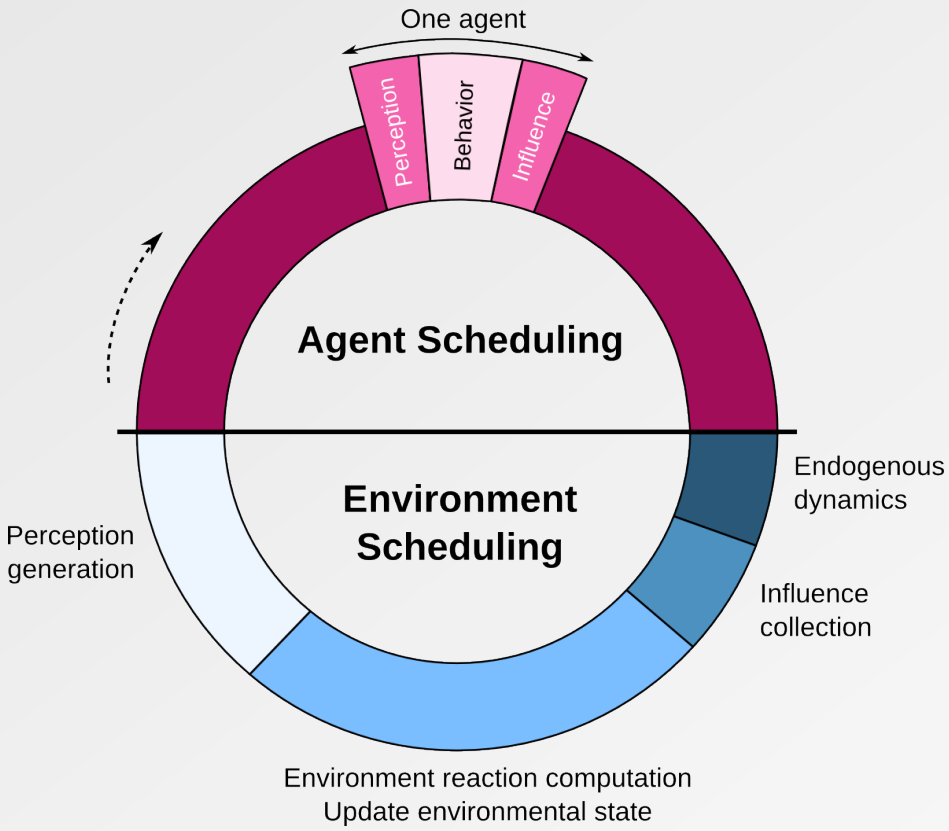
Body-Mind Separation [Béhé, 2014]



Communication Dimension of the Environment

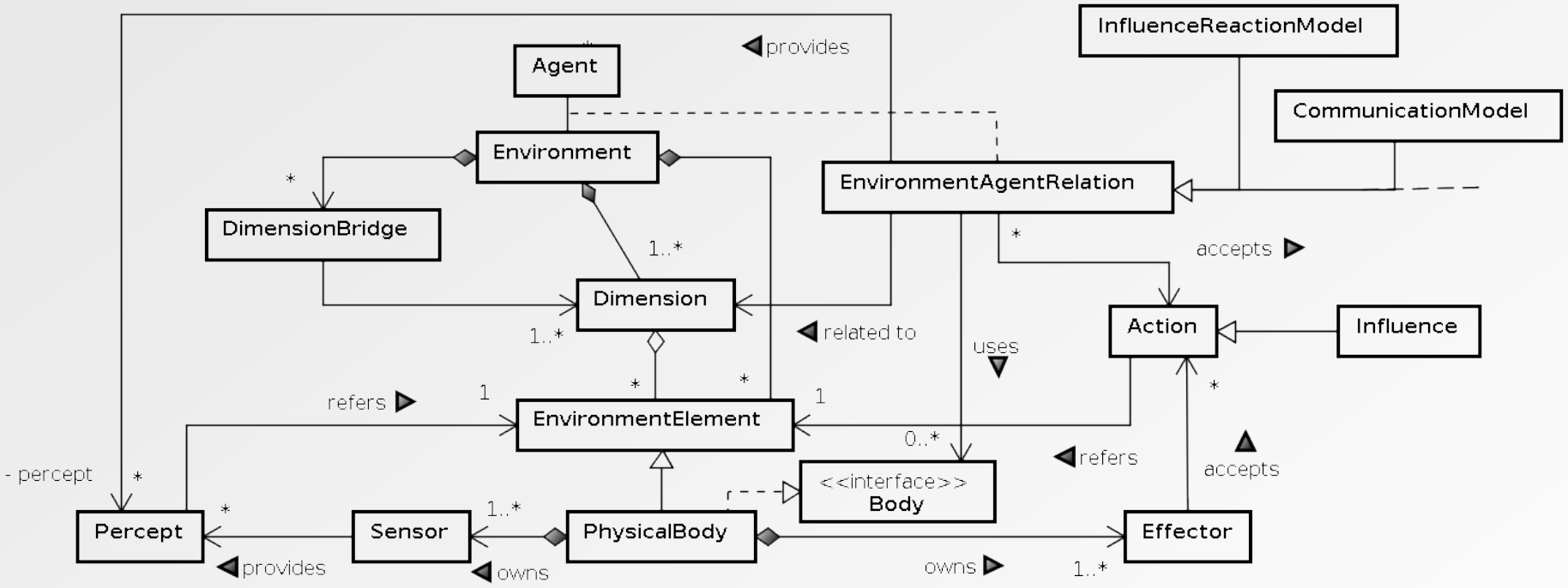
- Existing approaches:
  - Modeling for interaction: events, messages, etc.
  - Social relationships between agents: authoritative, auction, CNP, etc.
- Our approach:
  - Modeling based on the definition of a Space.
  - Use the SARL built-in default template : Event.
- Ability to set specific social environments:
  - FIPA
  - Organizational (MOISE, CRIO, AGR, etc.)

Simulation Lifecycle in the Environment Model: an example




```
behavior Environment {
  var time = 0

  on Initialize {

    every(500.milliseconds)

      (influences.size == participants.size
       || isTimeOut)

      => wake( new SimulationStep )

  }

  on SimulationStep {

    computeEndogenousInfluences
    computePhysicalReactionsFromInfluences

    time = time + 1

    computePhysicalPerceptions
    deliverPerceptions
    deliverMessages

  }
}
```

Model of the Environment



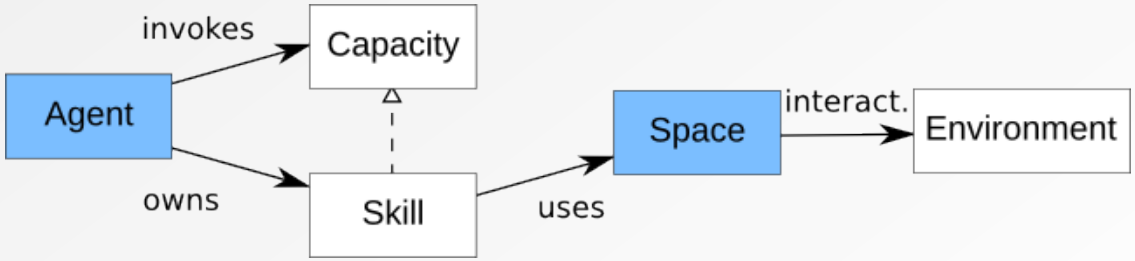
From Physical Environment to Agent: the Perception

- Perception of an agent: set of perceived objects
- Notification of perception changes with an event

```
event Perception {
  var objects : Percept[]
}
```

- Agent reacts to perception change:

```
agent A {
  on Perception {
    for (objectPerception : occurrence.objects) {
      /* Do something with the perceived object */
    }
  }
}
```



Capacities to Act in the Physical Dimension

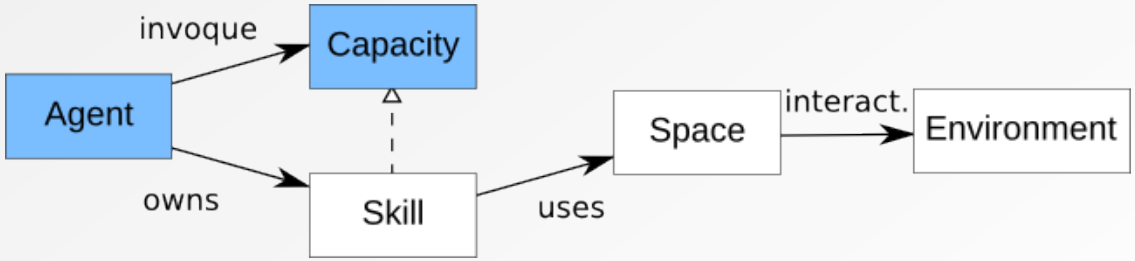
- Definition of the functions available to the agent:

```
capacity AbstractBodyCapacity {
  def getSpeed : double
  def killBody
}
capacity CarCapacity extends AbstractBodyCapacity {
  def getPosition : Pair<Segment, double>
  def getOrientation : Direction
  def accelerate(force : float)
  def turnLeft(angle : float)
  def turnRight(angle : float)
}
capacity PedestrianCapacity extends AbstractBodyCapacity {
  def getPosition : Point
  def getOrientation : Vector
  def move(force : Vector)
}
```

Abstract definition

For car driver

For pedestrian



Agent-Environment Relation : Influence-Reaction Model

- Agents do not directly change the state of the physical dimension.
- They emit influences of different types (motion, etc.) that are used for computing the dimension state's changes.

```
event Influence {
  var object : EObject
}

event Motion extends Influence {
  var linearSpline : double
  var linearShift : double
  var path : Segment[]
}
```

```
event Action extends Influence {
  var name : String
  var parameters : Object[]
}
```