SARL

Agent Programming Language

Sebastian Rodriguez - sebastian.rodriguez@rmit.edu.au

Summer Workshops 2026

Security and Resilience Hub

27 Feb 2026

Acknowledgement of Country

RMIT University acknowledges the people of the Woi wurrung and Boon wurrung language groups of the eastern Kulin Nation on whose unceded lands we conduct the business of the University.

RMIT University respectfully acknowledges their Ancestors and Elders, past and present.

RMIT also acknowledges the Traditional Custodians and their Ancestors of the lands and waters across Australia where we conduct our business.

Artwork 'Sentient' by Hollie Johnson
Hollie is a Gunaikurnai and Monero Ngarigo woman from Gippsland who graduated from RMIT with a BA in Photography in 2016.

Agenda

9:30 – 9:45	Introduction and housekeeping
9:45 – 10:15	Agents and their applications
10:15 – 11:15	Programming a SARL agent
11:15 – 11:45	Coffee Break
11:45 – 12:15	Interacting with the external world
12:15 – 12:45	Multiagent communications
12:45 – 1:15	LLM-Agents in SARL
1:15 – 1:30	Close and Survey

The Team

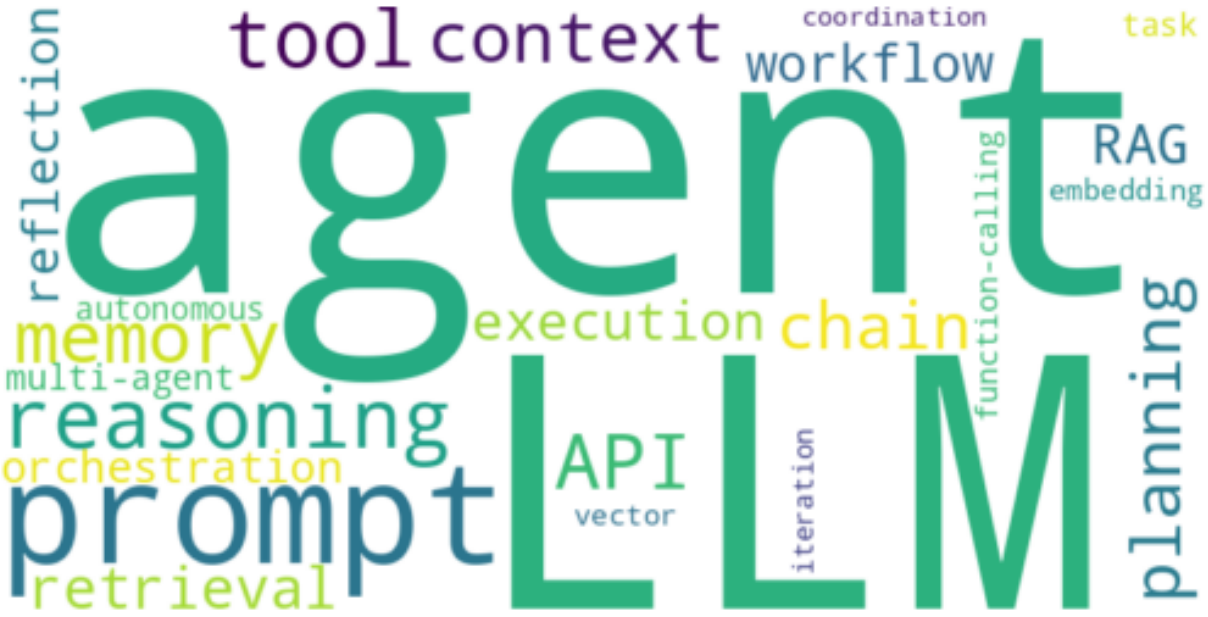
Sebastian Rodriguez - sebastian.rodriguez@rmit.edu.au

John Thangarajah – john.thangarajah@rmit.edu.au

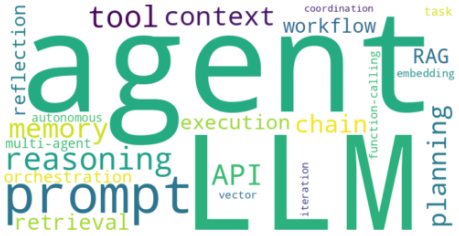
Bailey Vogt - s3906263@student.rmit.edu.au

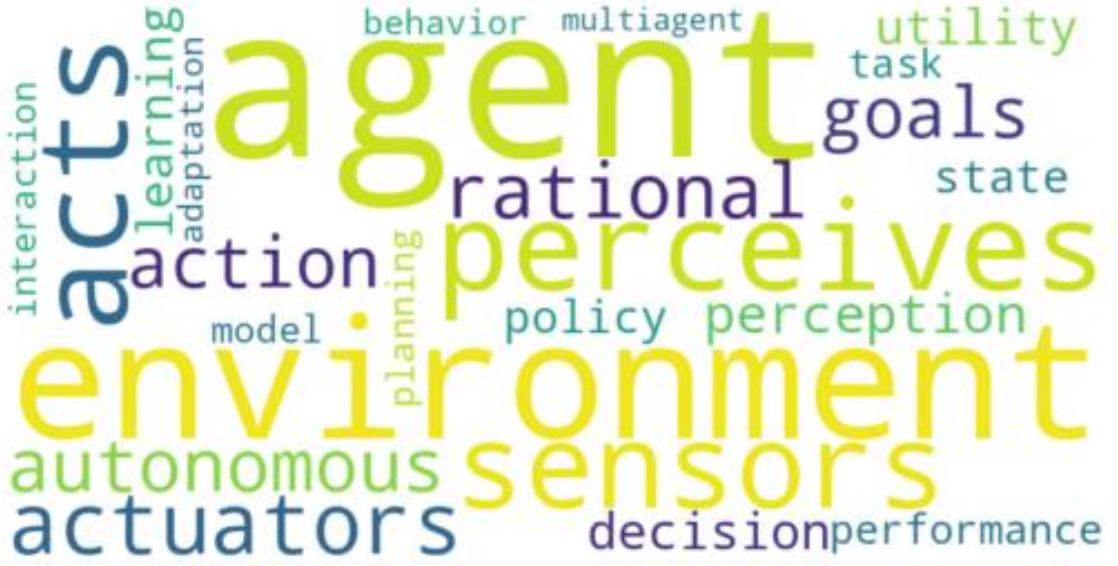
Andrew Davey - s3589434@student.rmit.edu.au

What is an Agent? - Terminology

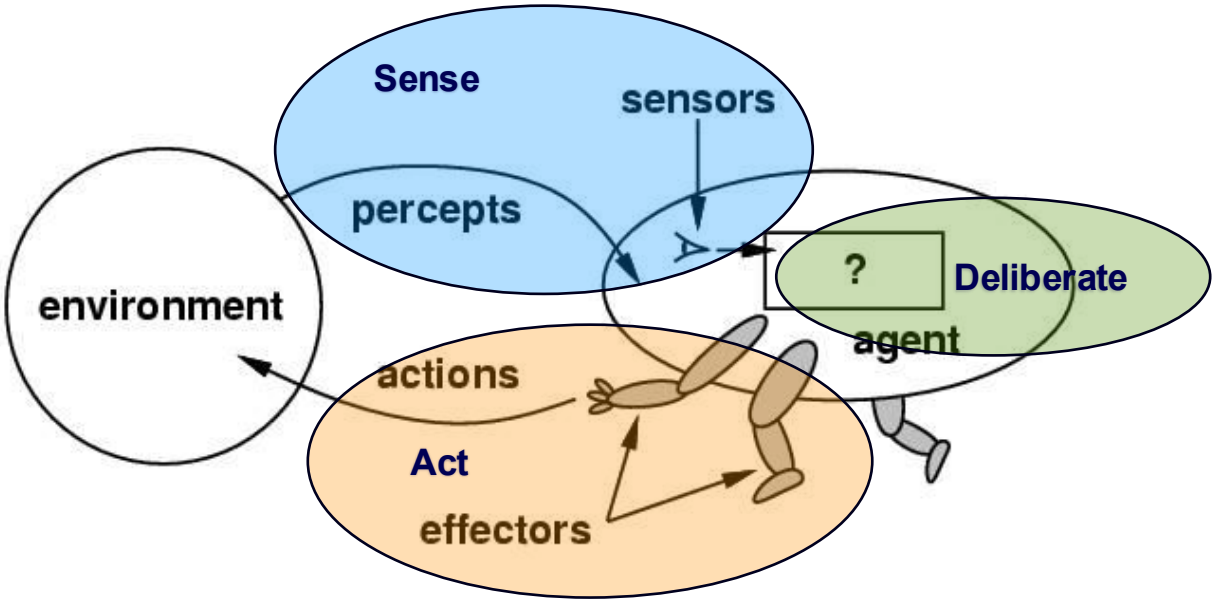


What is an Agent? - Terminology





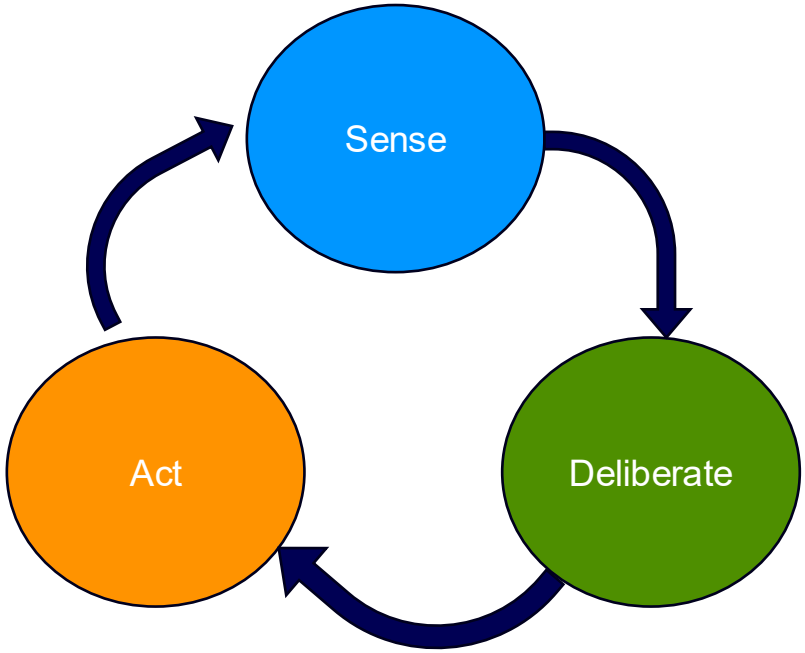
What is an Agent? - Terminology



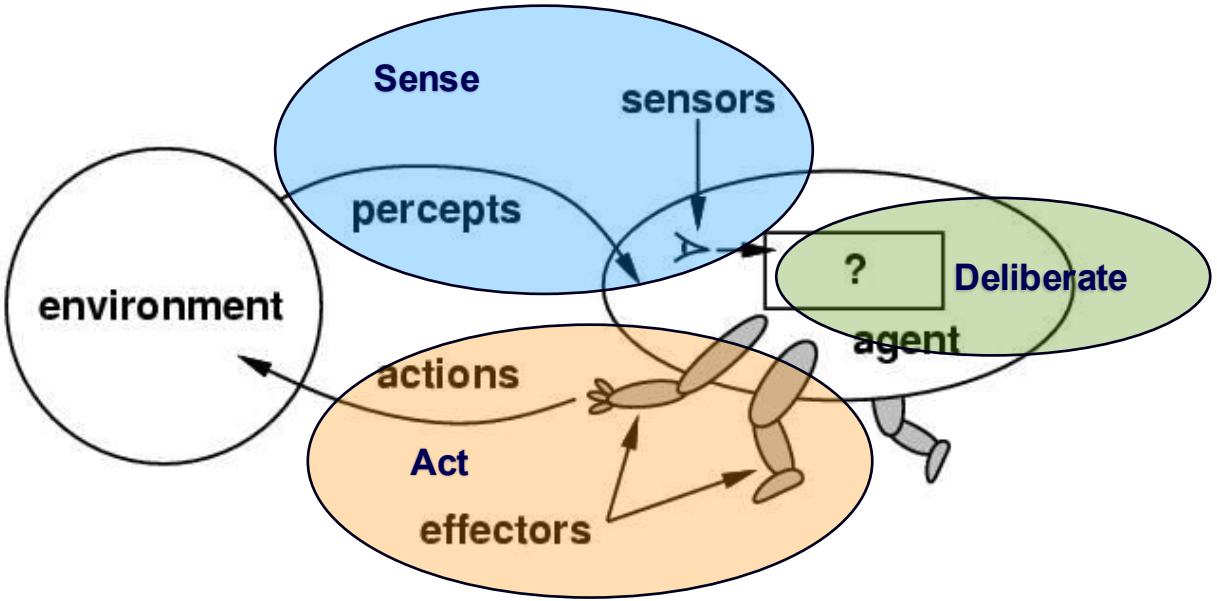
An **agent** is anything that can be viewed as perceiving its environment through **sensors** and acting upon that environment through **actuators**.

— Russell & Norvig, AI: A Modern Approach (2010)

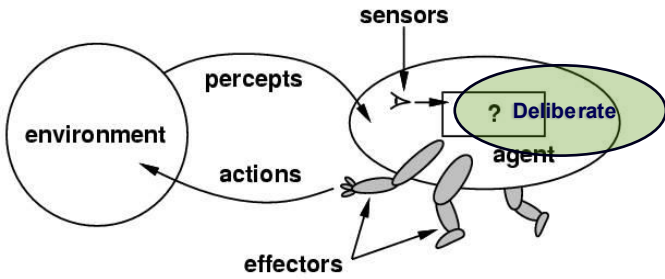
What is an Agent? - Terminology

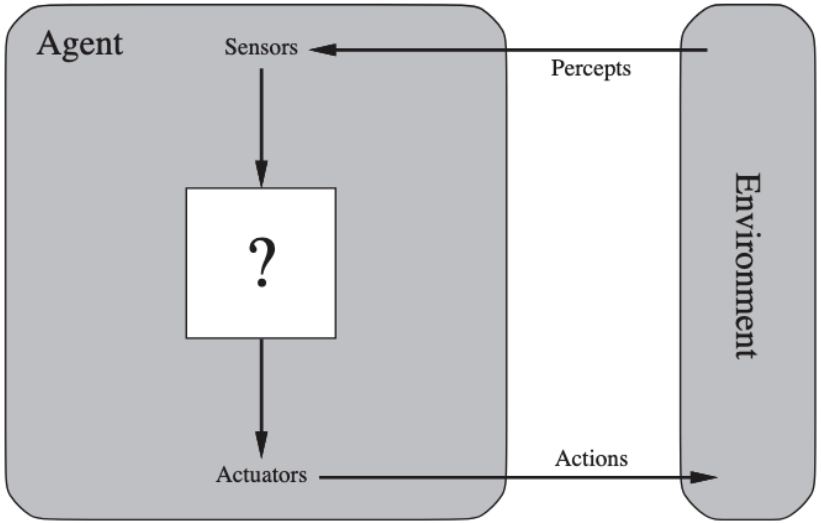


What is an Agent? - Terminology



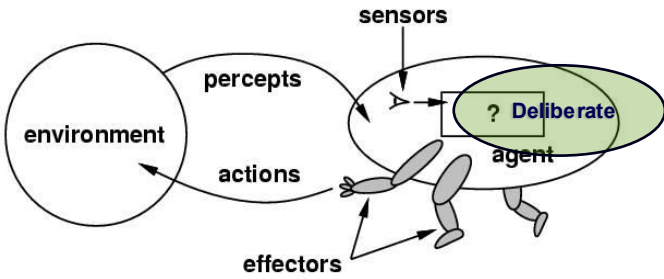
What is an Agent? - Deliberation

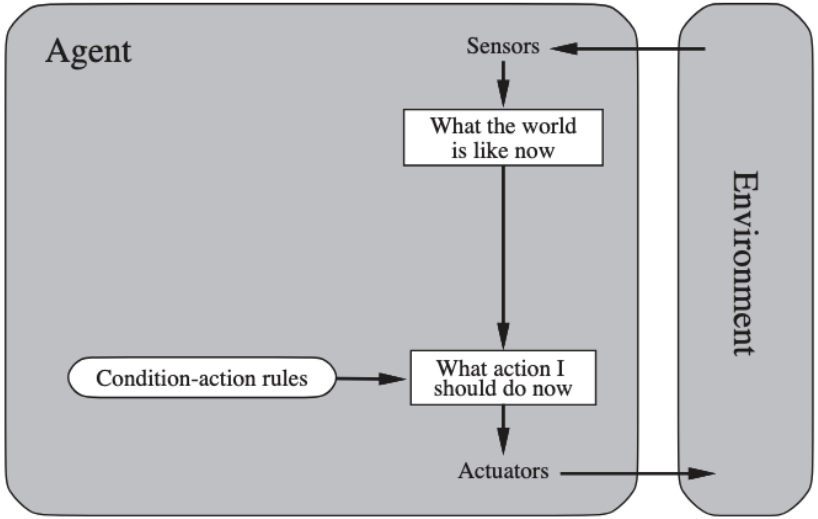




Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norvig

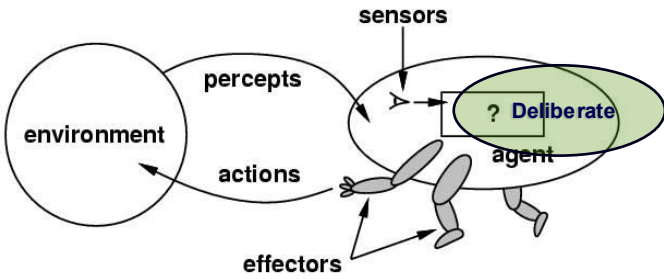
Deliberation - Reflex

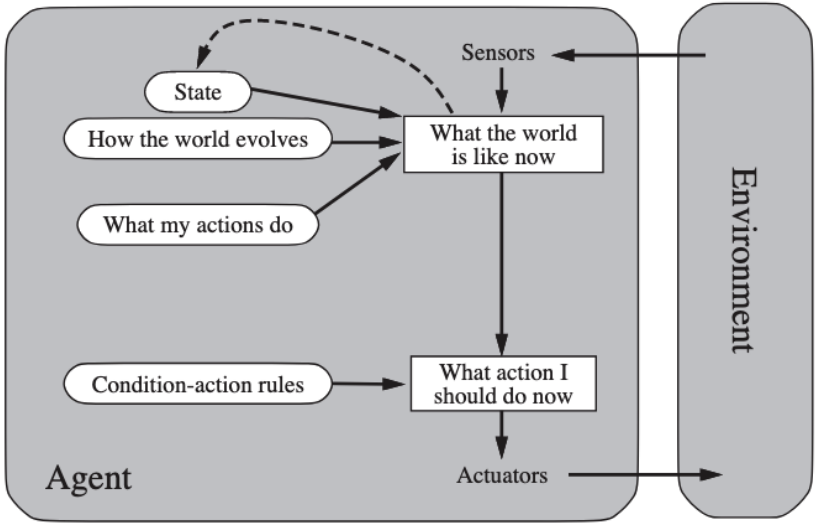




if *temperature* > *22* **then** *turnACOn*

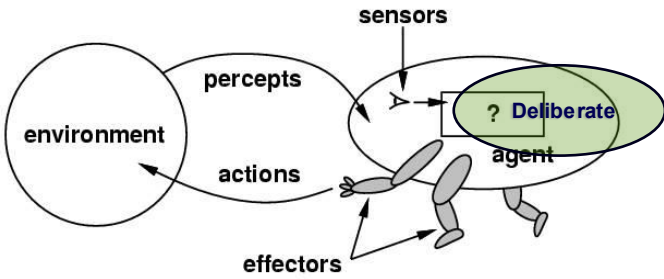
Deliberation – Model based Reflex

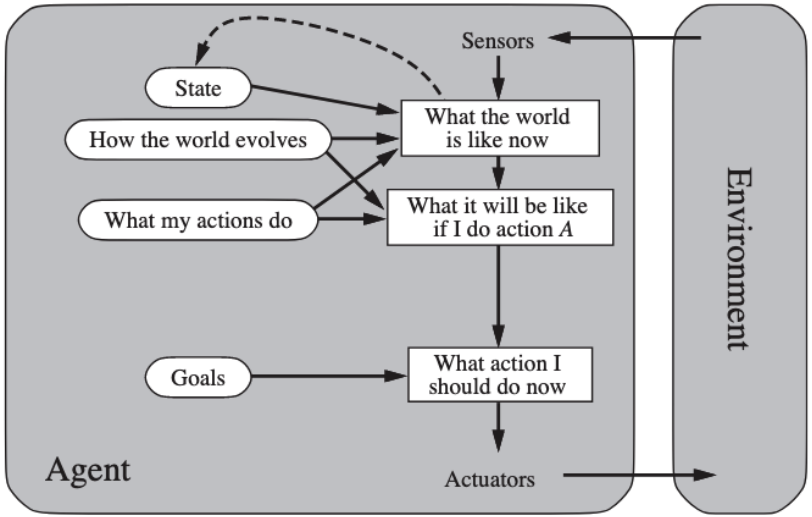


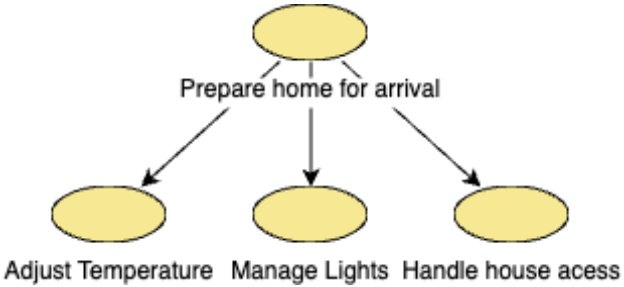


if *temperatureIsRising* **then** *turnACOn*

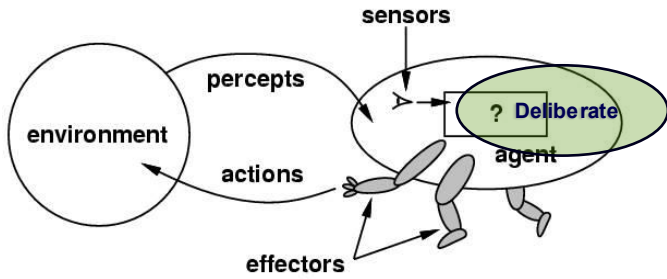
Deliberation – Goal Based

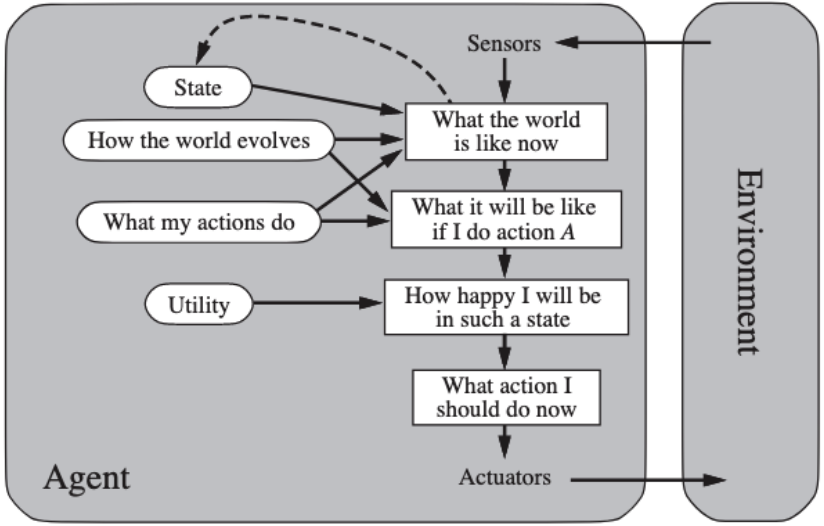






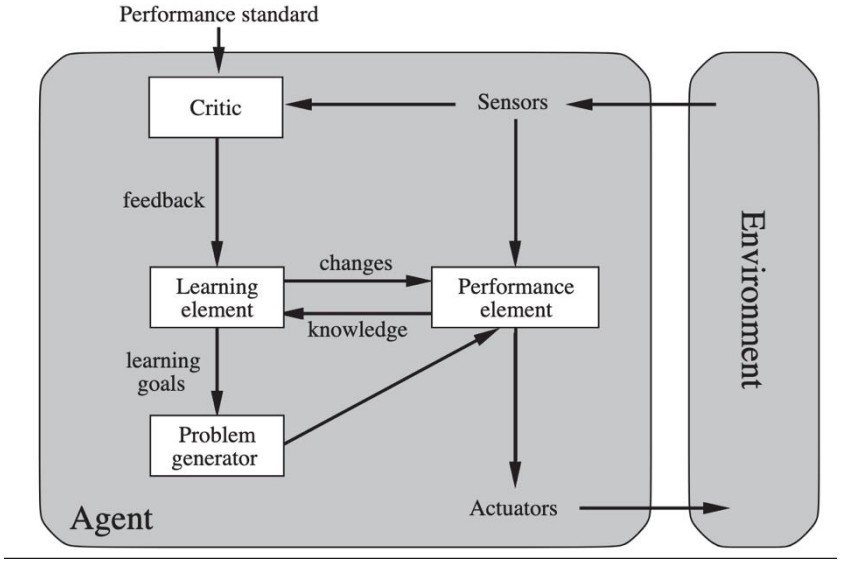
Deliberation – Utility

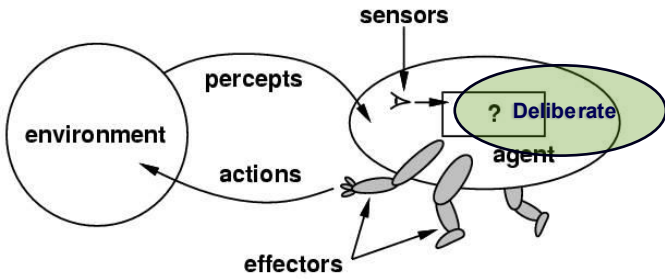




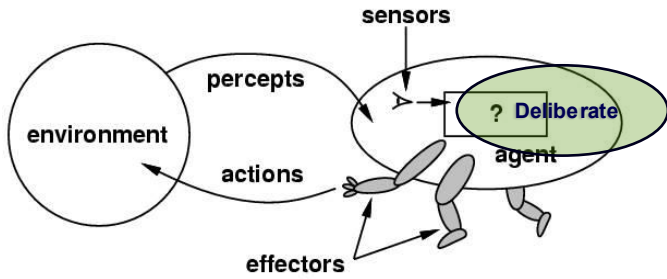
What do I prefer,
lower the temperature or save energy?

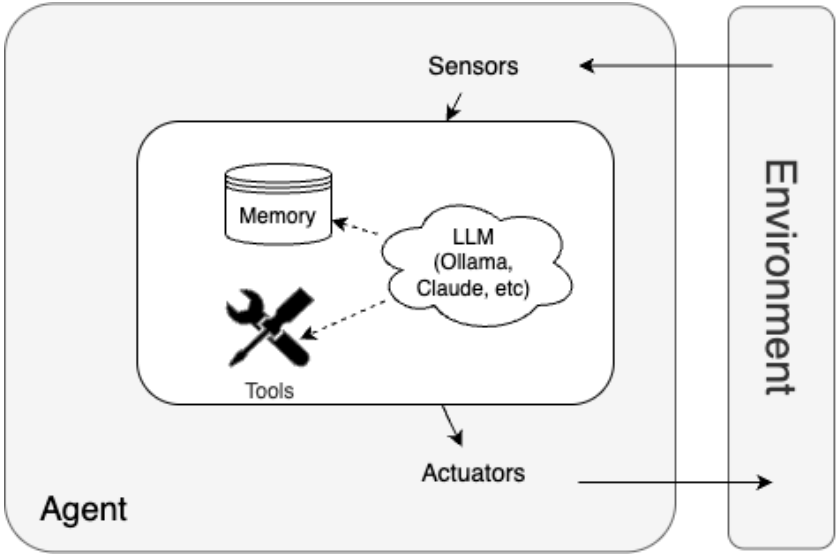
Deliberation – Learning





Deliberation – LLM



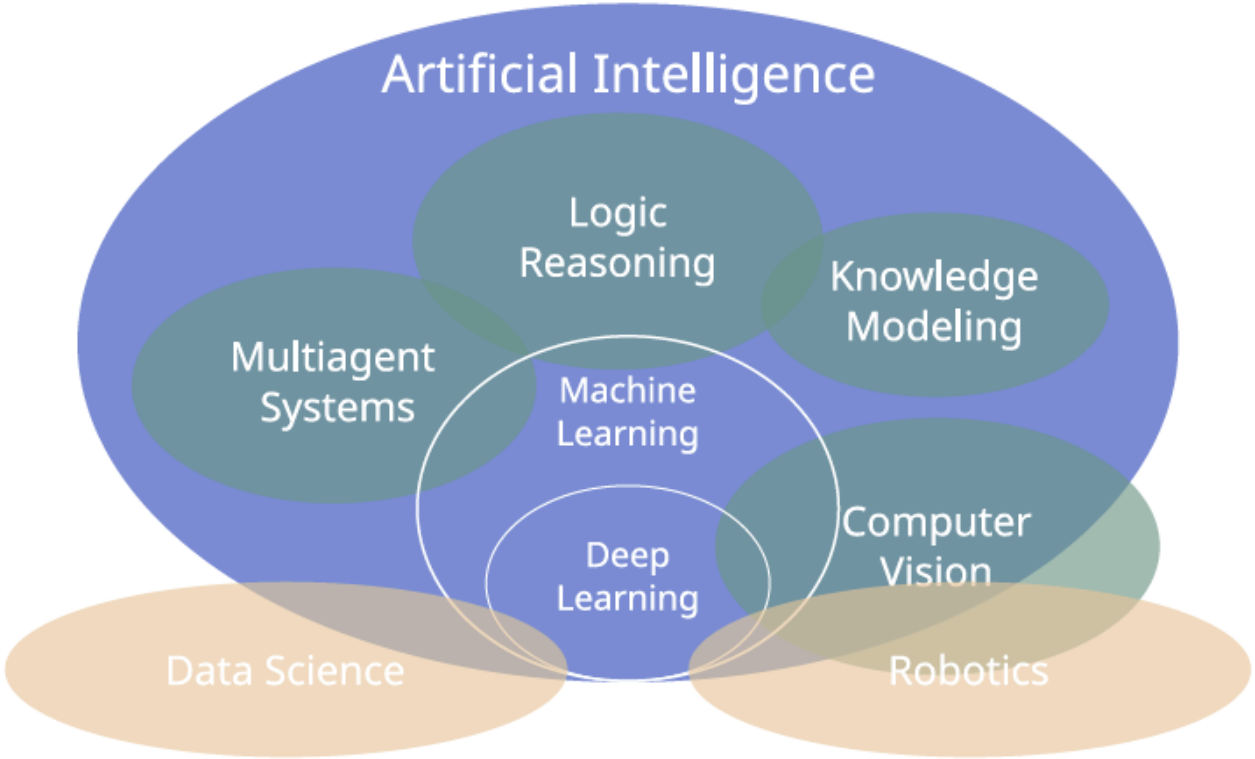


Use the LLM to Decide what to do

- LLMs are hosted outside of the agent
- Tools can be actuators

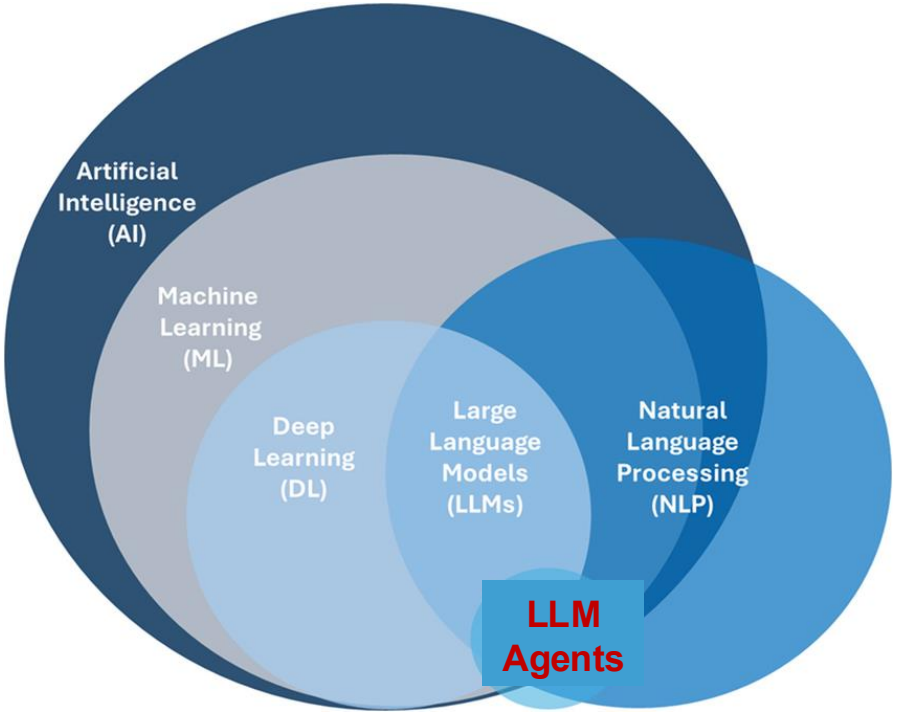
Not in Russell & Norvig

Agents and AI



Based on ACM Classification

LLM-Agents



How AI Transforms Regulatory Submission: Current Clinical Implementation and Future Prospects. Podichetty et al, 2025

Domains

Autonomous Vehicles & Robotics

Healthcare

IoT and Industry 4.0

Gaming

Digital Twins and Simulations

Customer Service and Virtual Assistants

Smart Buildings

…

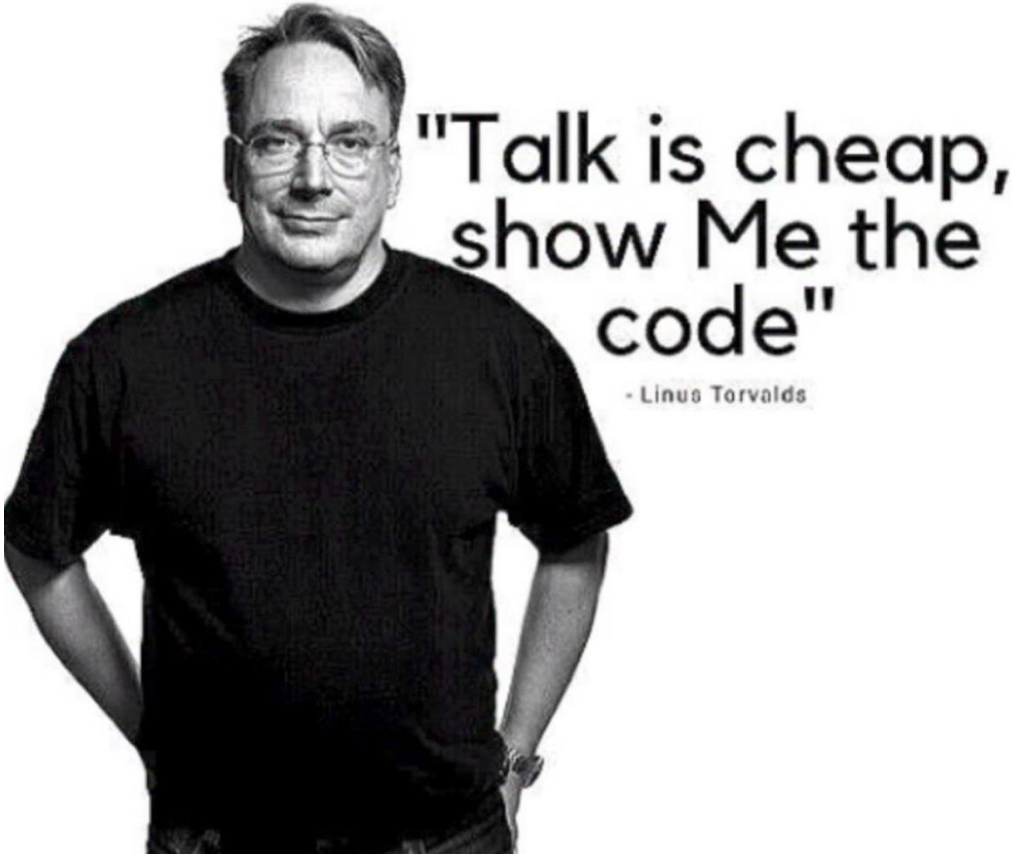
…

Use Cases

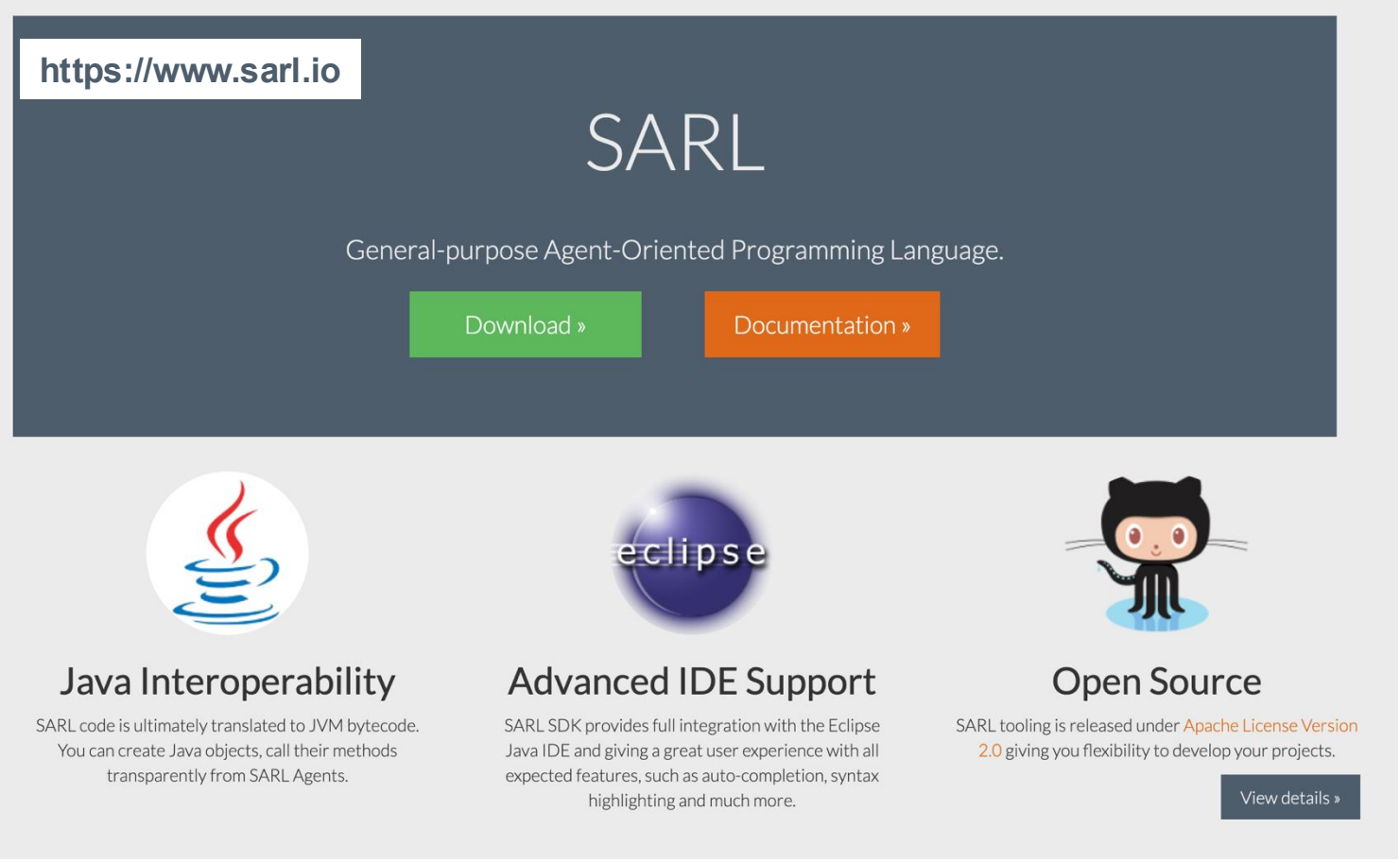
Use Case	Key Requirements	Example Domains	Best Type of Agent	Why LLM-Agents Are Weak Here
Real-Time Safety-Critical Systems	Determinism; millisecond response; certifiable logic; safety guarantees	Air traffic control; space systems; industrial automation	Reflex agents; Hybrid reactive–deliberative agents	Unbounded latency; non-deterministic outputs; difficult to formally verify
Formal Planning & Optimal Control	Completeness; optimality; constraint satisfaction; cost minimization	Logistics optimization; mission planning; supply chain scheduling	Goal-oriented agents; Classical planning agents	No guarantees of optimality or constraint satisfaction; approximate reasoning
Multiagent Coordination & Strategic Reasoning	Commitment tracking; protocol compliance; joint intention reasoning; norm enforcement	Smart grids; energy markets; traffic coordination; defense simulations	BDI agents; Norm-aware agents	No explicit belief/desire/intention model; weak protocol compliance; poor equilibrium reasoning
Embedded / Resource-Constrained Systems	Low latency; minimal memory footprint; energy efficiency; local autonomy	IoT devices; edge robotics; automotive ECUs; sensor networks	Reflex agents; Lightweight goal-based agents	High computational cost; cloud dependency; energy inefficient
High-Frequency Trading & Finance	Deterministic execution; microsecond-level response; auditable strategy logic	Algorithmic trading; quantitative finance systems	Reactive agents; Utility agents	Latency too high; stochastic outputs unacceptable; weak explainability

Use Cases

Use Case	Key Requirements	Example Domains	Better Type of Agent	Why Non-LLM Agents Are Weak Here
Natural Language Enterprise Copilot for Policy & Operational Support	Advanced natural language understanding; handling ambiguity; contextual reasoning across large document corpora;; conversational interaction; adaptive clarification	Government policy advisory assistants; internal knowledge copilots;	LLM-Agent with RAG + tool orchestration	Classical agents require explicit symbolic representations and predefined ontologies; brittle under ambiguous or underspecified input; high engineering cost to model open-ended language; poor scalability across heterogeneous document sources

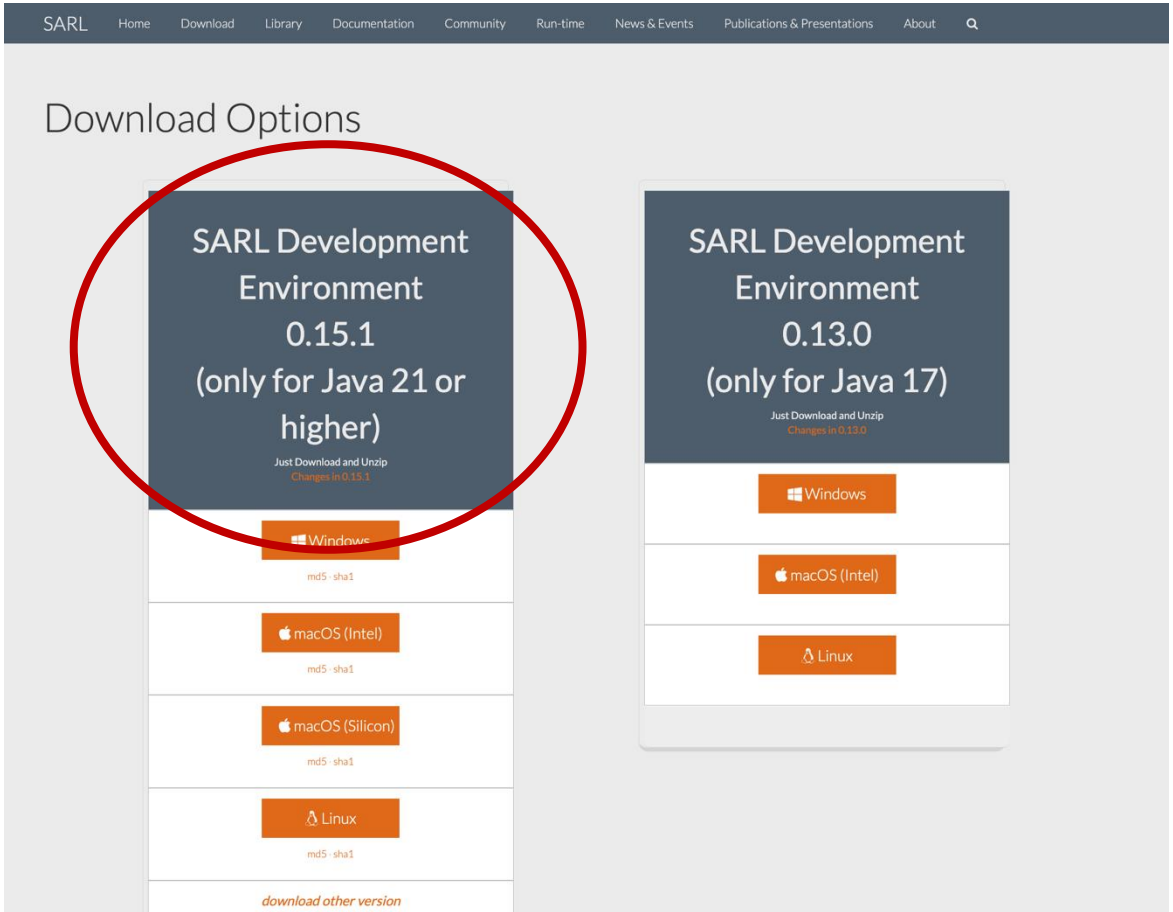

"Talk is cheap,
show Me the
code"
- Linus Torvalds

SARL – Set up your IDE



SARL – Set up your IDE

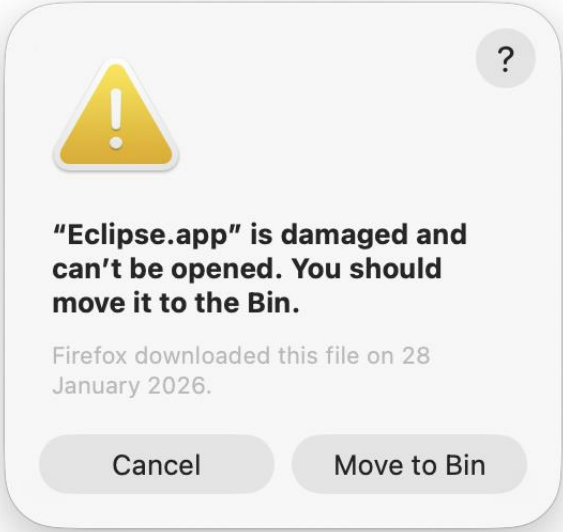
https://www.sarl.io/download/



Mac Users

Source:

typically, due to macOS security features (Gatekeeper)

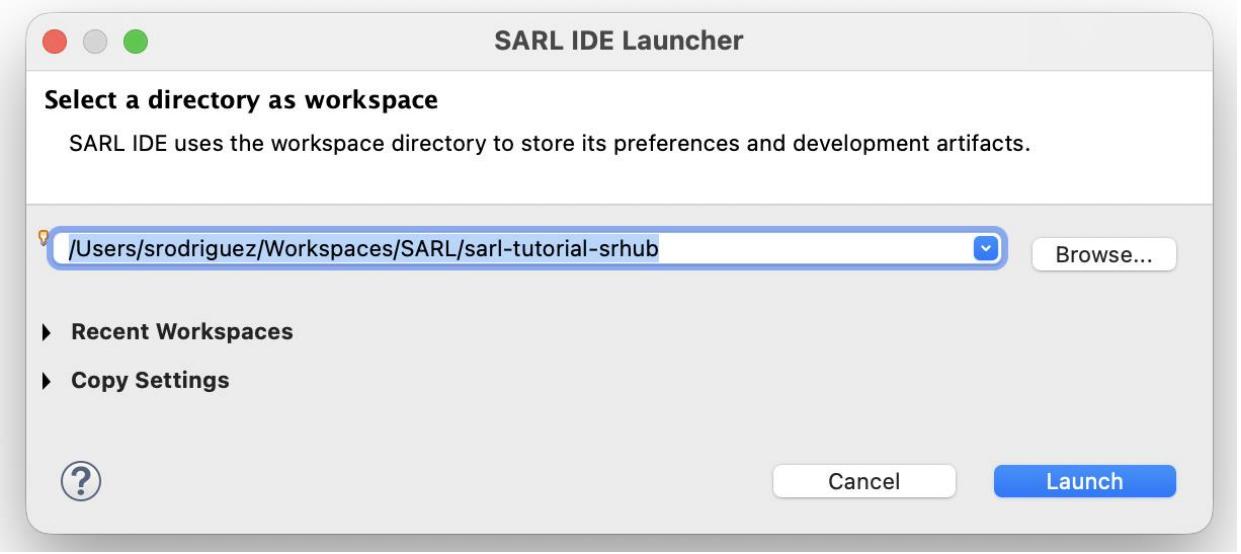


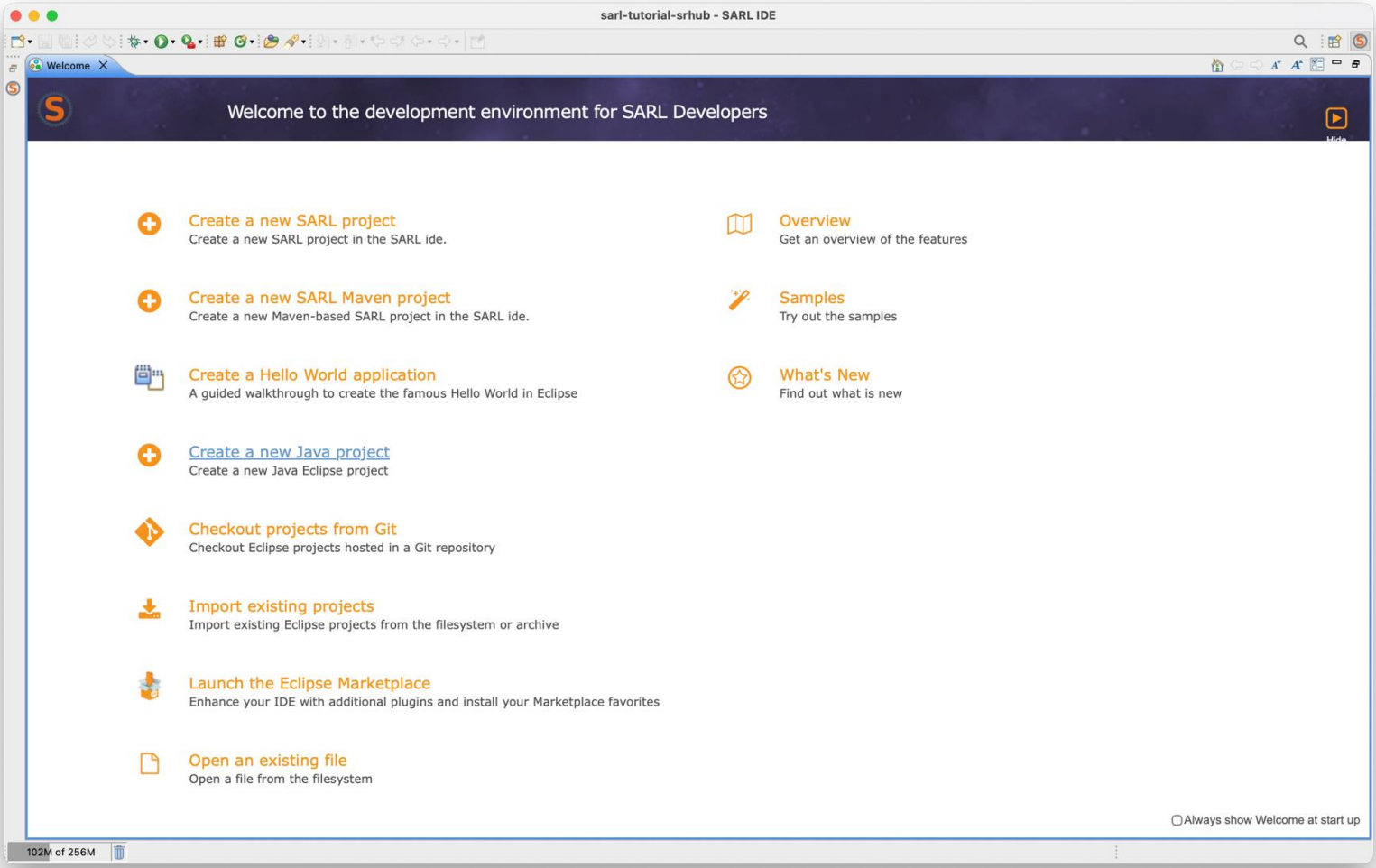
Solution:

In your terminal

$ xattr -d com.apple.quarantine /Applications/Eclipse.app

Choose where to place your projects (aka your Workspace)




sarl-tutorial-srhub - SARL IDE
Welcome
Welcome to the development environment for SARL Developers
Create a new SARL project
Create a new SARL project in the SARL ide.
Create a new SARL Maven project
Create a new Maven-based SARL project in the SARL ide.
Create a Hello World application
A guided walkthrough to create the famous Hello World in Eclipse
Create a new Java project
Create a new Java Eclipse project
Checkout projects from Git
Checkout Eclipse projects hosted in a Git repository
Import existing projects
Import existing Eclipse projects from the filesystem or archive
Launch the Eclipse Marketplace
Enhance your IDE with additional plugins and install your Marketplace favorites
Open an existing file
Open a file from the filesystem
Overview
Get an overview of the features
Samples
Try out the samples
What's New
Find out what is new
Always show Welcome at start up
102M of 256M

Access the code

https://github.com/srodriguez-research/sarl-tutorial

SARL – Essential Concepts

Goal

Provide the fundamental abstractions

Individual

Agent architecture-agnostic

Powerful (yet simple) extension mechanism

Social

Multiple interaction mechanisms

Distribution (network) abstraction

Collective

Native support for "recursive agents" (holons)

Open-Source Project

Apache v2 License

http://www.sarl.io

http://www.github.com/sarl

Modern Tooling and Ecosystem

Full IDE Support

- Debugging
- Autocompletion
- Error Checking
- much more…

Compatible with modern tools

- Maven / Docker (Deploy)
- Junit / Cucumber (Testing)

Java interoperability

- Easy integration with other systems
- Databases
- Simulators
- APIs / Web services
- IoT Devices

SARL – Users

Open-Source Project
Apache v2 License
http://www.sarl.io
http://www.github.com/sarl

SARL – Multiagent system

A collection of agents interacting together in a collection of shared distributed spaces

Dimensions of a MAS

Individual: the Agent abstraction (Agent, Capacity, Skill)
Social: the Interaction abstraction (Space, Event, etc.)
Collective: the Holon abstraction (Context)

Main Concepts

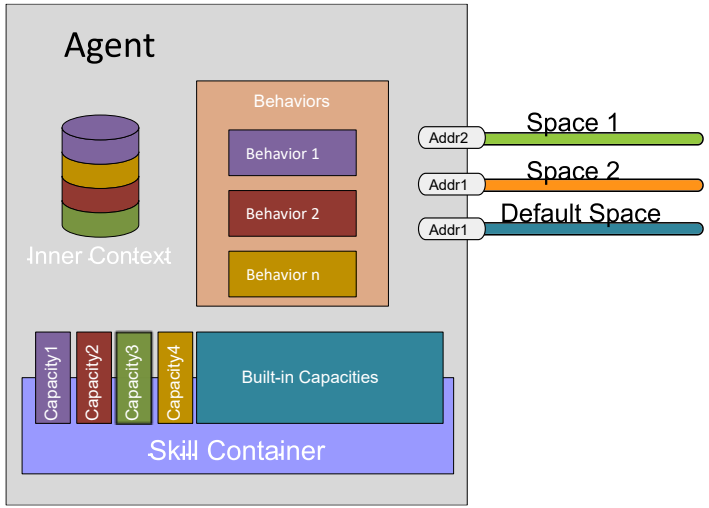
Agent
Capacity / Skill
Space
Context

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

http://www.sarl.io

Agent

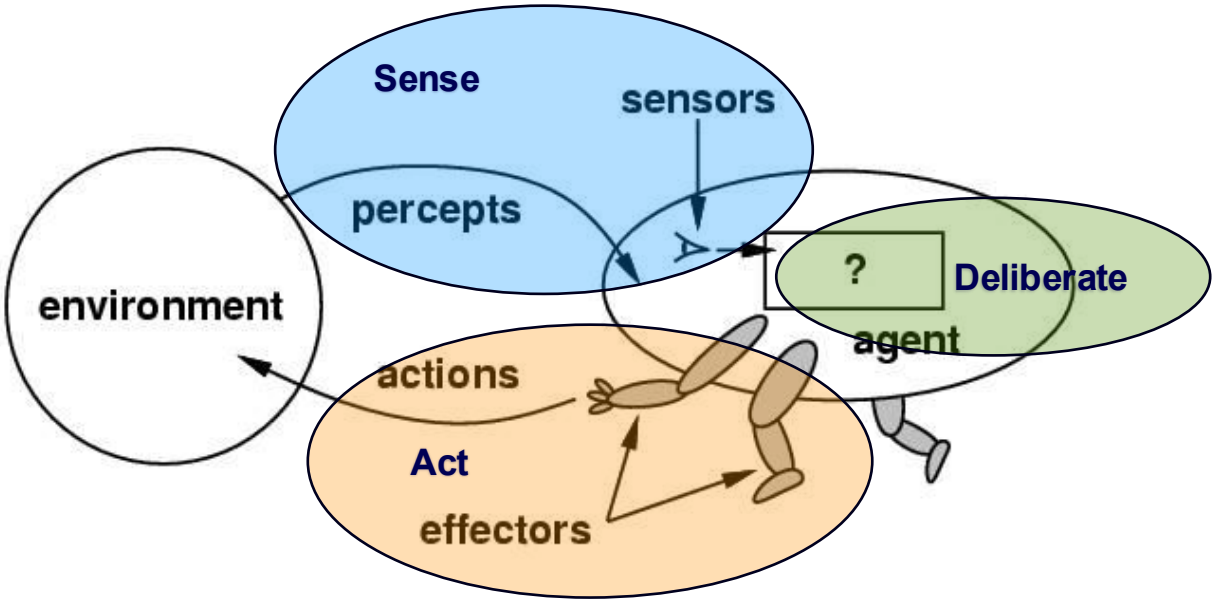
- An agent is an autonomous entity having some intrinsic skills to implement the capacities it exhibits.
- An agent initially owns native capacities called Built-in Capacities.
- An agent defines a Context.




```
agent HelloAgent {
  on Initialize {
    println("Hello World!")
  }
  on Destroy {
    println("Goodbye World!")
  }
}
```

Simple Aircon Controller Agent

If temperature is 22 C or higher, turn the AC on

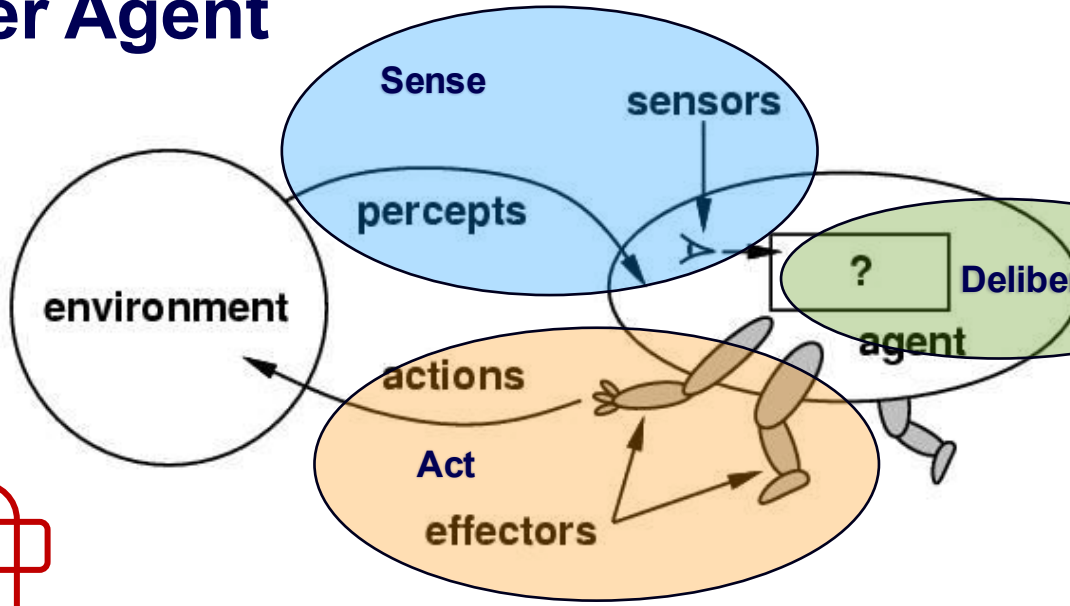


Simple Aircon Controller Agent

Percepts definition

```
event TemperatureChanged {
    var temperature: int
}

agent AirconController {

    on TemperatureChanged{
        if(occurrence.temperature > 22){
            //turn Aircon On
        }else{
            //turn Aircon Off
        }
    }
}
```

Sense

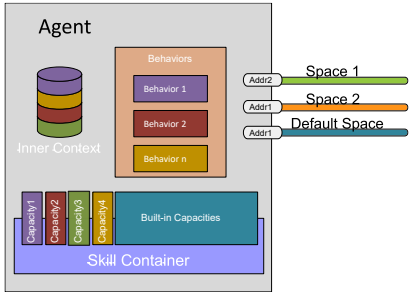


Simple Aircon Controller Agent

```
event TemperatureChanged {
   var temperature: int
}

agent AirconController2 {

   on TemperatureChanged [temperature > 22] {
       // turn Aircon On
   }

   on TemperatureChanged [temperature <= 22] {
       // turn Aircon Off
   }

}
```

```
event TemperatureChanged {
   var temperature: int
}

agent AirconController {

   on TemperatureChanged{
       if(occurrence.temperature > 22){
           //turn Aircon On
       }else{
           //turn Aircon Off
       }
   }
}
```

Capacity and Skill



Action

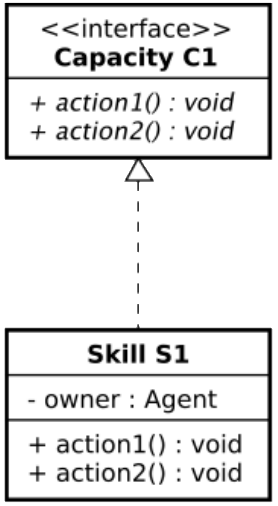
- A specification of a transformation of a part of the designed system or its environment.
- Guarantees resulting properties if the system before the transformation satisfies a set of constraints.
- Defined in terms of pre- and post-conditions.

Capacity

- Specification of a collection of actions.

Skill

- A possible implementation of a capacity fulfilling all the constraints of its specification, the capacity.



Enables the separation between a generic behaviour and agent-specific capabilities.

Example

```
capacity Logger {
    def debug(msg : String)
    def info(msg : String)
}

skill ConsoleLogger implements Logger {

    def debug(msg : String) {
        System.out.println("DEBUG: "+msg)
    }

    def info(msg : String) {
        System.out.println("INFO : " + msg)
    }

}
```

```
agent SimpleAgent {
    uses Logger

    on Initialize {
        setSkillIfAbsent(new ConsoleLogger, Logger)
        info("Hello world")
        debug("This is a debug message")
    }
}
```

Built-in Capacities

A SARL Agent has inherently a set of Built-in Capacities

Behaviors	Manage sub-behaviors
Lifecycle	Agent life management (e.g. Agent spawning and killing)
Schedules	Scheduling of tasks
ExternalContextAccess	Access to external context
InnerContextAccess	Access to inner context
DefaultContextInteractions	Access to the default context
Logging	Access to the logging mechanism associated to the agent.

Some useful actions from built-in capacities

Behaviours

wake(evt: Event) – emit an event as a new perception for the agent

Logging

info; debug; warning, etc.

Lifecycle

killMe – stops agent execution

spawn(Agent, params) - Spawns a new Agent inside the default context of this agent.

Schedule

in(delay: long, proc: Procedure) – Schedule a given task to be executed after the specified delay.

every (period: long, proc:Procedure) - Schedule a periodic execution of the given task

Example

```
event Say {
    var msg : String
}

agent SayEventAgent {
    uses Behaviors, Logging, Schedules, Lifecycle

    on Initialize{
        wake(new Say => [msg="Hello World!"])
    }

    on Say{
        info("Reacting to a Say Event")

        info(occurrence.msg)

        in(2.seconds)[killMe]
    }

    on Destroy{
        info("Goodbye World")
    }
}
```

```
[INFO, 4:39:03pm, AGENT-88405bae-a979-4b49-994e-79a6ee50ebaa] Reacting to a Say Event
[INFO, 4:39:03pm, AGENT-88405bae-a979-4b49-994e-79a6ee50ebaa] Hello World!
[INFO, 4:39:05pm, AGENT-88405bae-a979-4b49-994e-79a6ee50ebaa] Goodbye World
[INFO, 4:39:05pm, SARL Run-time Environment] Stopping the kernel services
[INFO, 4:39:05pm, SARL Run-time Environment] All kernel services are stopped
```

Behaviours

- Behavior maps a collection of perceptions represented by events to a sequence of Actions.
- Enable to split the global behaviour of an agent into smaller behaviours ⇒ modularity

```
behavior TheBehavior {
  on Initialize {
    println("Hello!")
  }
  on Destroy {
    println("Goodbye!")
  }
}
```

Behaviour Example

```
capacity ACControl{
    def turnOn
    def turnOff
}

behavior AirconController {
    uses ACControl

    on TemperatureChanged [temperature > 22] {
        turnOn
    }

    on TemperatureChanged [temperature <= 22] {
        turnOff
    }
}

agent SmartHomeController {
    uses Behaviors
    on Initialize {
        registerBehavior(new AirconController(this))
    }

}
```

What we covered so far

- Create SARL Project / Convert to Maven Project
- Understanding the IDE
- Create a simple HelloWorld Agent
- Understanding Standard Events
- Add Capacities to your Agent
  - Schedules
  - Lifecycle
- Debugging your Agent
- Event Driven programming

Remember this?

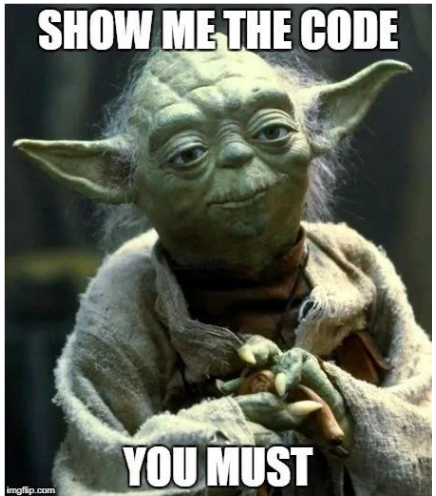
```
event Say {
    val msg : String
}

agent SayAgent {
    uses Behaviors, Logging, Schedules, Lifecycle
    var count : int = 0

    on Initialize {
        wake(new Say("Hello World! " + count))
    }

    on Say {
        info("Reacting to a Say Event")
        info(occurrence.msg)
        in(2.seconds)[killMe]
    }

    on Destroy {
        info("Goodbye World")
    }
}
```

```
[INFO, 4:39:03pm, AGENT-88405bae-a979-4b49-994e-79a6ee50ebaa] Reacting to a Say Event
[INFO, 4:39:03pm, AGENT-88405bae-a979-4b49-994e-79a6ee50ebaa] Hello World!
[INFO, 4:39:05pm, AGENT-88405bae-a979-4b49-994e-79a6ee50ebaa] Goodbye World
[INFO, 4:39:05pm, SARL Run-time Environment] Stopping the kernel services
[INFO, 4:39:05pm, SARL Run-time Environment] All kernel services are stopped
```

Say 5 times

Adapt the example before so it says “Hello world” 5 times.

```
[INFO, 8:14:34pm, AGENT-997003d1-7301-4293-a576-8c6554441e41] Hello World! 0
[INFO, 8:14:35pm, AGENT-997003d1-7301-4293-a576-8c6554441e41] Hello World! 1
[INFO, 8:14:36pm, AGENT-997003d1-7301-4293-a576-8c6554441e41] Hello World! 2
[INFO, 8:14:37pm, AGENT-997003d1-7301-4293-a576-8c6554441e41] Hello World! 3
[INFO, 8:14:38pm, AGENT-997003d1-7301-4293-a576-8c6554441e41] Hello World! 4
[INFO, 8:14:38pm, AGENT-997003d1-7301-4293-a576-8c6554441e41] Goodbye World
[INFO, 8:14:39pm, SARL Run-time Environment] Stopping the kernel services
[INFO, 8:14:39pm, SARL Run-time Environment] All kernel services are stopped
```



Say 5 times - option with if

```
agent SayTimesAgent {
   uses Behaviors, Logging, Schedules, Lifecycle
   var count : int = 0

   on Initialize{
       wake(new Say("Hello World! "+count))
   }

   on Say {
       count+=1
       info(occurrence.msg)
       if (count > 4) {
           killMe
       } else {
           in(1.seconds)[wake(new Say( "Hello World! " + count))]
       }
   }

   on Destroy{
       info("Goodbye World")
   }
}
```

Say 5 times - option with events

```
agent SayTimesEventsAgent {
   uses Behaviors, Logging, Schedules, Lifecycle

   var count : int = 0

   on Initialize {
       wake(new Say( "Hello World! " + count))
   }

   on Say[count<5] {
       count += 1
       info(occurrence.msg)
       in(1.seconds)[wake(new Say("Hello World! " + count))]
   }

   on Say [count >= 5] {
       killMe
   }

   on Destroy {
       info("Goodbye World")
   }
}
```

Create your own Agent : Factorial

Create an Agent that can calculate the factorial of a number using events

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 3 \times 2 \times 1$$
$$= \begin{cases} 1, & \text{if } n = 0 \\ n \times (n-1)!, & \text{if } n \geq 1. \end{cases}$$

For example,

$$5! = 5 \times 4! = 5 \times 4 \times 3 \times 2 \times 1 = 120.$$

Solution: https://www.sarl.io/library/io.sarl.eclipse.examples.factorial/index.html

Preparing the second part

Install the open llama from https://ollama.com/

Once installed, in your terminal

$ ollama serve

$ ollama pull llama3.2

Let’s take 30 minutes to talk agents!

Photo by Frank Leuderalbert on Unsplash

Access External Services

Java interoperability

- Easy integration with other systems
- Databases
- Simulators
- APIs / Web services
- IoT Devices

If there is JAVA SDK or API, SARL Agents can access use them

Access External Services

Weather Agent

Use https://open-meteo.com/

Try in locally in your terminal:

curl -s "https://api.open-meteo.com/v1/forecast?latitude=-37.814&longitude=144.9633&current=temperature_2m"

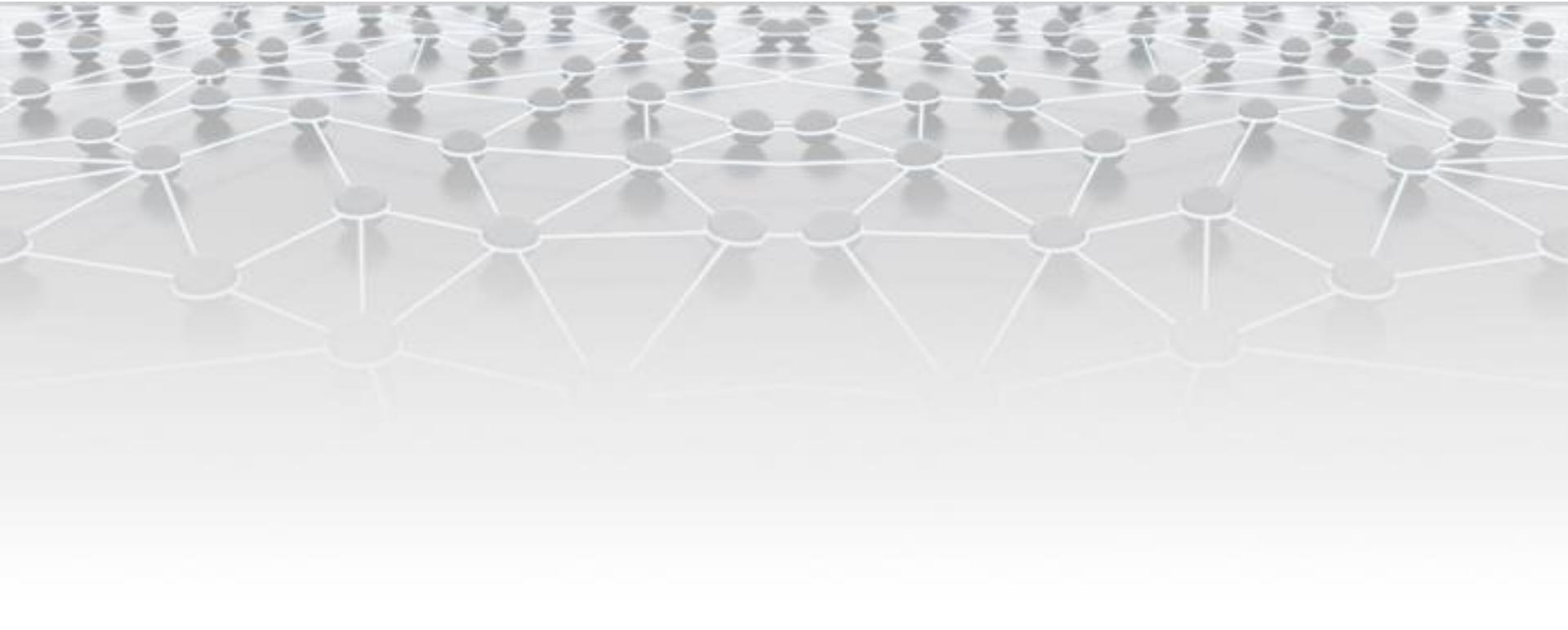

```
{
  "latitude": -37.75,
  "longitude": 145.0,
  "generationtime_ms": 0.029325485229492188,
  "utc_offset_seconds": 0,
  "timezone": "GMT",
  "timezone_abbreviation": "GMT",
  "elevation": 19.0,
  "current_units": {
    "time": "iso8601",
    "interval": "seconds",
    "temperature_2m": "°C"
  },
  "current": {
    "time": "2026-02-25T04:30",
    "interval": 900,
    "temperature_2m": 20.9
  }
}
```


Weather Agent –

```
agent WeatherAgent {
   uses Logging, Lifecycle

   val API_ENDPOINT = "https://api.open-meteo.com/v1/forecast?latitude=-37.814&longitude=144.9633&current=temperature_2m"

   on Initialize {
       val client = HttpClient.newHttpClient();
       val request = HttpRequest.newBuilder().uri(URI.create(API_ENDPOINT)).GET().build();
       val response = client.send(request, HttpResponse.BodyHandlers.ofString());

       // Check the status code and body
       info("Status Code: " + response.statusCode());
       info("Response Body: " + response.body());

       val objectMapper = new ObjectMapper();
       val rootNode = objectMapper.readTree(response.body());

       info("Current temperature is {0}", rootNode.get("current").get("temperature_2m"))

       killMe

   }
}
```

Can you make this a shareable Capacity / Skill?

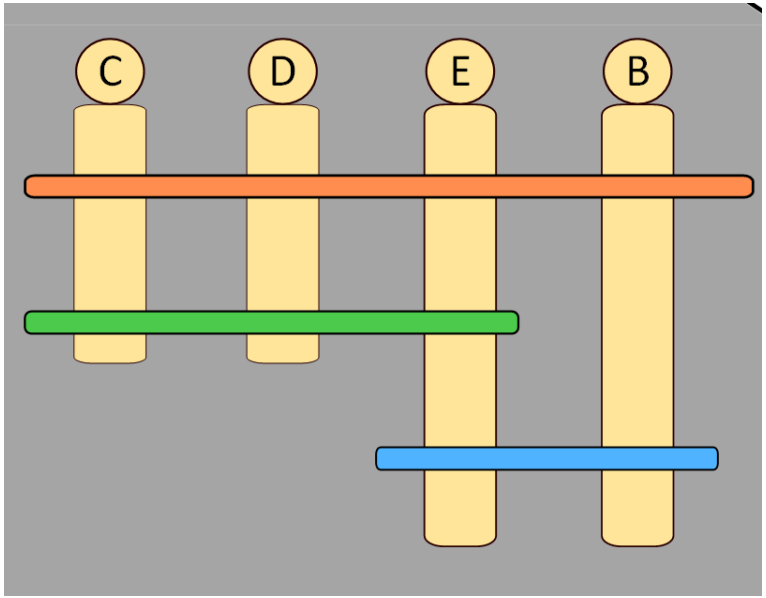
Multi Agent Systems



The more, The merrier

Multiagent system (in SARL)

A collection of agents interacting together in a collection of shared distributed spaces



Communication Support

Space

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

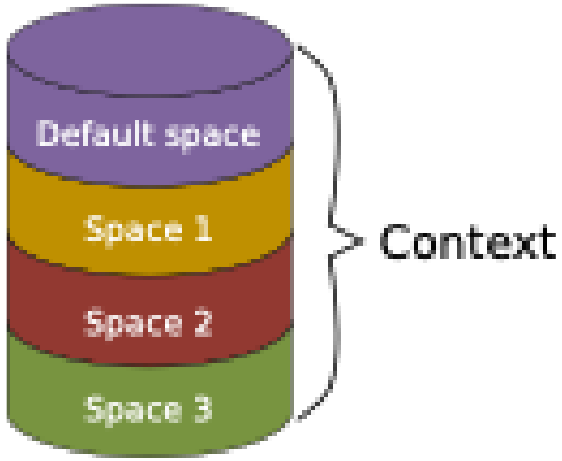
Space Specification

- 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.

Context and Spaces

Context

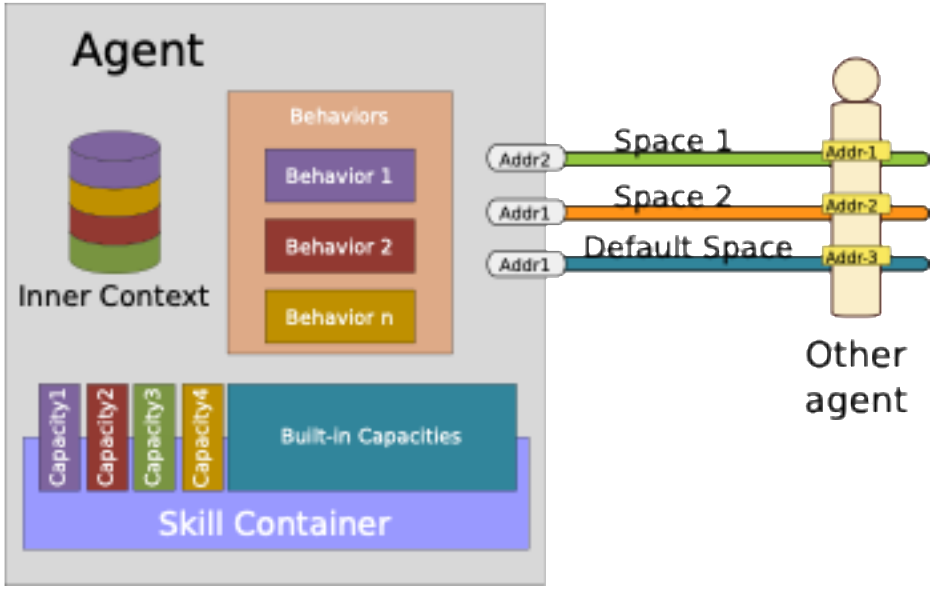
- Defines the boundary of a sub-system.
- Collection of Spaces.
- Every Context has a Default Space.
- Every Agent has a Default Context, the context where it was spawned.



Default Space

Default Space: an Event Space

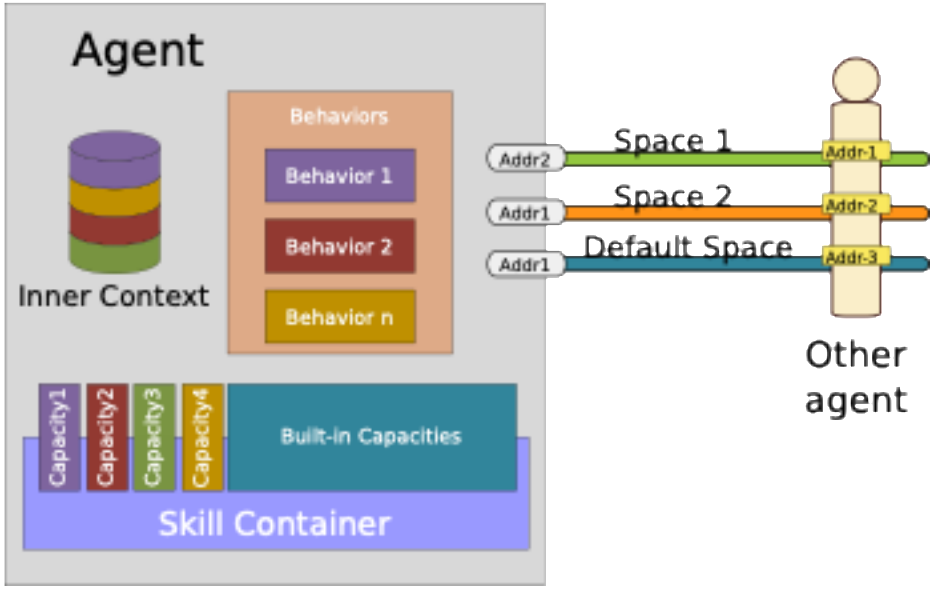
- Event-driven interaction space.
- Default Space of a context, contains all agents of the considered context.
- Event: the specification of some occurrence in a Space that may potentially trigger effects by a participant.



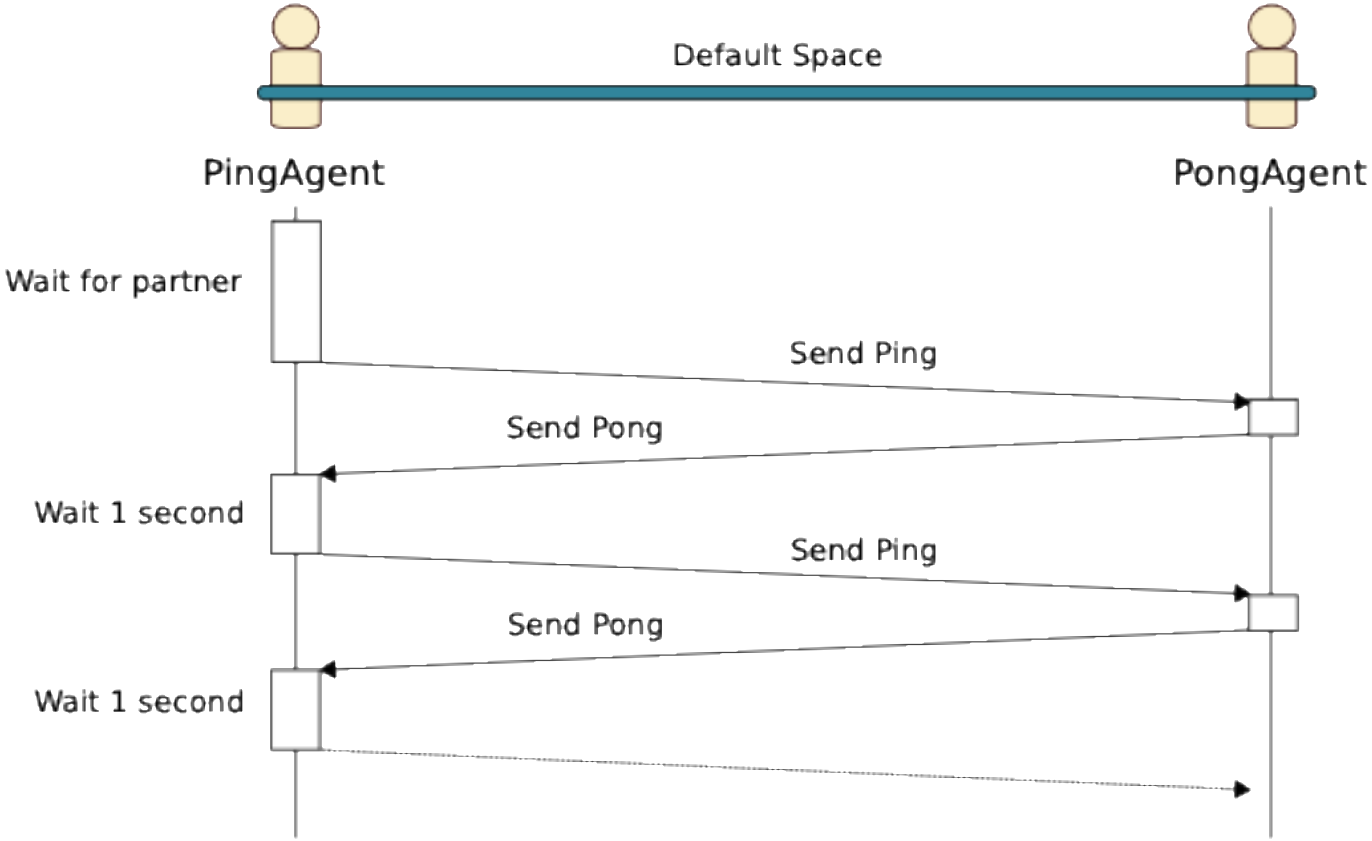
Default Space

Default Space: an Event Space

- Agents can emit events
- Agents perceive events they register for using an “on Event” clause
- Events can carry data



Communicating with other agents



Communicating with other agents

```
event Ping {
    val index : int
}


event Pong {
    val index : int
}
```

Communicating with other agents

```
agent PingAgent {

    uses DefaultContextInteractions, Schedules, Logging

    on Pong {
        info("Receiving Pong #" + occurrence.index)
        info("Sending Ping #" + (occurrence.index + 1))
        emit(new Ping( occurrence.index + 1 )) [it == occurrence.source]
    }

    on Initialize {
        info("Waiting for Pong Agent")
        val taskVar = task("waiting_for_partner")

        taskVar.every(1000) [
            if (defaultSpace.numberOfStrongParticipants > 1) {
                info("Pong Agent detected.")
                info("Sending Ping #0")
                emit( new Ping(0) )
                taskVar.cancel
            }
        ]
    }

}
```

```
event Ping {
    val index : int
}


event Pong {
    val index : int
}
```

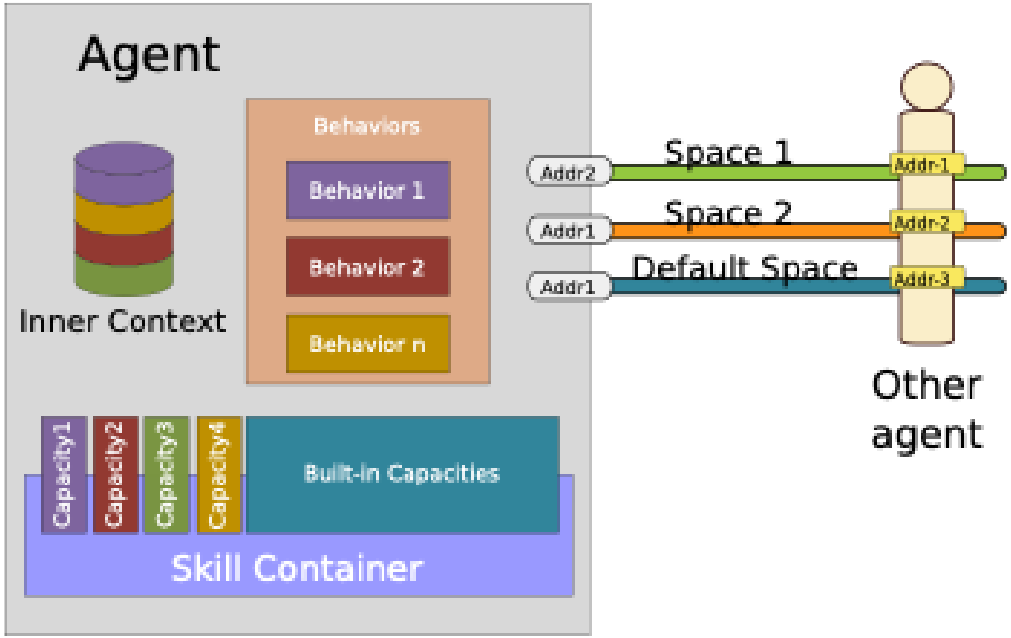
Communicating with other agents

```
agent PingAgent {

    uses DefaultContextInteractions, Schedules, Logging

    on Pong {
        info("Receiving Pong #" + occurrence.index)
        info("Sending Ping #" + (occurrence.index + 1))
        emit(new Ping( occurrence.index + 1 )) [it == occurrence.source]
    }

    on Initialize {
        info("Waiting for Pong Agent")
        val taskVar = task("waiting_for_partner")

        taskVar.every(1000) [
            if (defaultSpace.numberOfStrongParticipants > 1) {
                info("Pong Agent detected.")
                info("Sending Ping #0")
                emit( new Ping(0) )
                taskVar.cancel
            }
        ]
    }

}
```

```
event Ping {
    val index : int
}


event Pong {
    val index : int
}
```

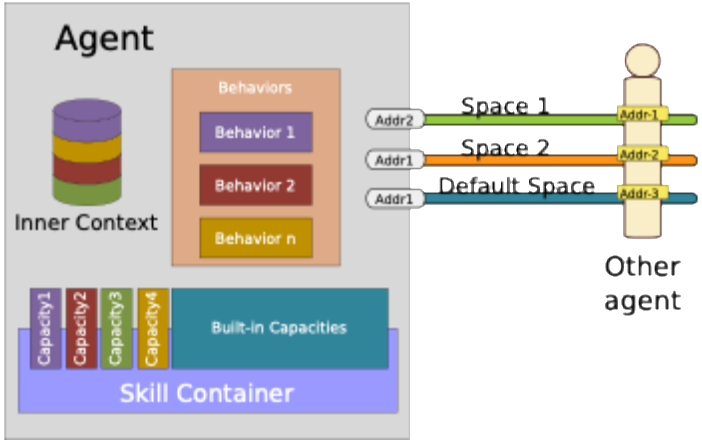
```
agent PongAgent {
    uses DefaultContextInteractions, Logging
    on Ping {
        info("Receiving Ping #" + occurrence.index)
        info("Sending Pong #" + occurrence.index)
        emit(new Pong( occurrence.index )) [it == occurrence.source]
    }

}
```

Creating Your own spaces



Creating Your own spaces

```
event Task {
    val name : String
}

agent SpaceAgent {
    uses DefaultContextInteractions, Behaviors, Logging, Lifecycle, Schedules, ExternalContextAccess

    on Initialize {
        val id = UUID.randomUUID
        val spaze = defaultContext.getOrCreateSpaceWithSpec(OpenEventSpaceSpecification, id)
        spaze.registerStrongParticipant(asEventListener)

        spawn(WorkerAgent, "W1", id)
        spawn(WorkerAgent, "W2")

        every(1.seconds) [
            spaze.emit(new Task("In task Space"))
            defaultSpace.emit(new Task("In Default Space"))
        ]
    }
}

agent WorkerAgent {
    uses DefaultContextInteractions, Behaviors, Logging

    var name : String

    on Initialize {
        name = occurrence.parameters.get(0) as String
```



Problems | Console | Tasks | Package Explorer

```
<terminated> SpaceAgent [SARL Agent] /Library/Java/JavaVirtualMachines/temurin-21.jdk/Contents/Home/bin/java (24 Feb 2026, 12:06:59 pm – 12:0
[INFO, 12:07:02pm, AGENT-b366057f-1270-4f2b-98d2-3eb2658722c6] [W1] Received Task In task Space
[INFO, 12:07:02pm, AGENT-b366057f-1270-4f2b-98d2-3eb2658722c6] [W1] Received Task In Default Space
[INFO, 12:07:02pm, AGENT-e2d4b890-8aa8-4721-8853-a45be7fed0ec] [W2] Received Task In Default Space
[INFO, 12:07:03pm, AGENT-b366057f-1270-4f2b-98d2-3eb2658722c6] [W1] Received Task In task Space
[INFO, 12:07:03pm, AGENT-b366057f-1270-4f2b-98d2-3eb2658722c6] [W1] Received Task In Default Space
[INFO, 12:07:03pm, AGENT-e2d4b890-8aa8-4721-8853-a45be7fed0ec] [W2] Received Task In Default Space
```

Challenges in agent communications (in General)

- Define the communication support (how to send data) – Solve by SRE
- Define the correct ”envelop” (how to find and address agents) – Solve by SARL
- Define the data to exchange (what to you need to send)
- Define the semantics of data (how to interpret the data of the message)
- Define the protocol of communication (what are the constraints when exchanging messages – order, timeouts, etc.)
- Monitoring
- Governance
- …

Large Language Models and SARL

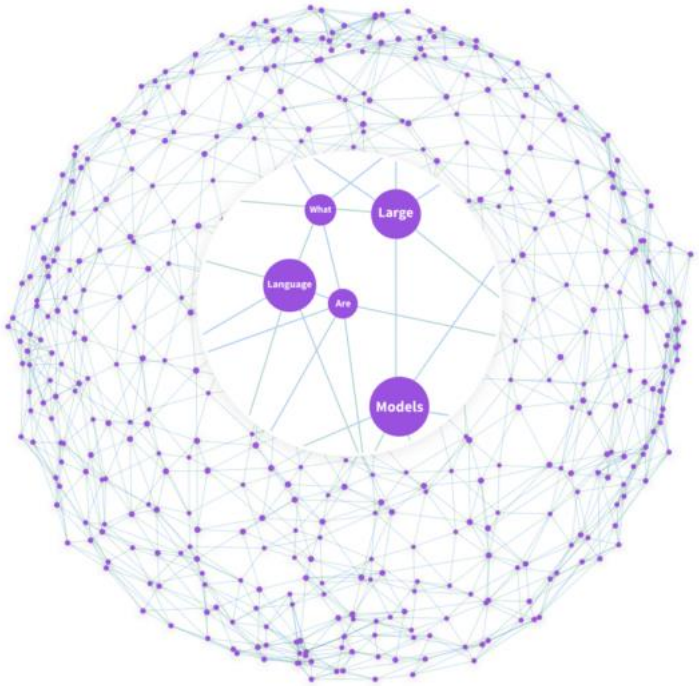


Image source: https://www.dataquest.io

Open LLama

Install from https://ollama.com/

Once installed, in your terminal

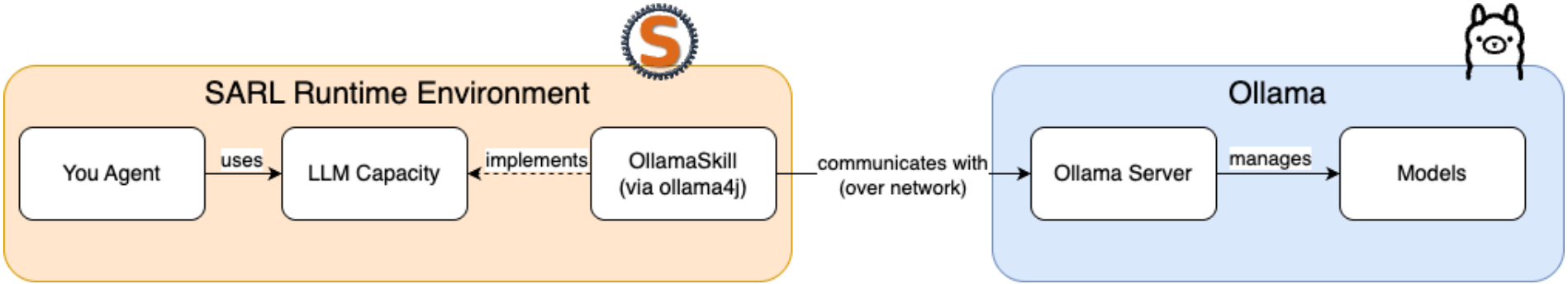
$ ollama serve

$ ollama pull llama3.2

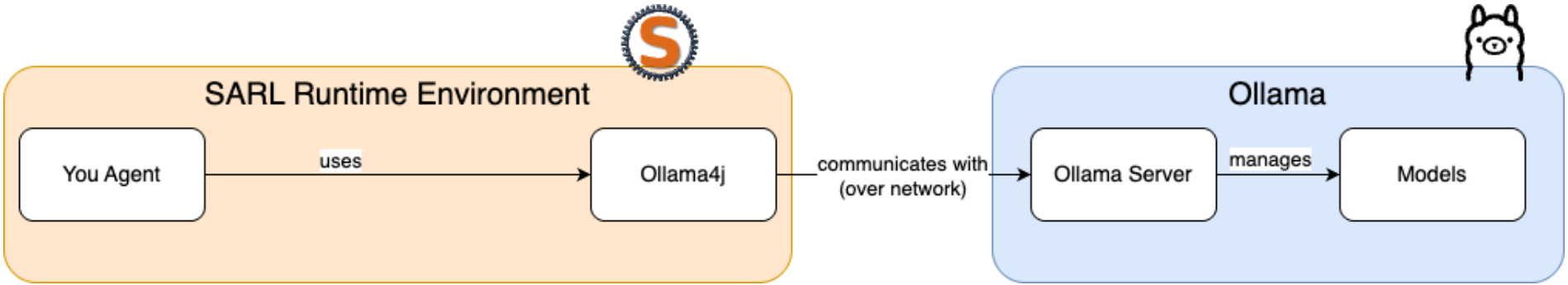
In your SARL project pom.xml file, add the following:

```
<dependencies>
        ...
        <dependency>
                <groupId>io.github.ollama4j</groupId>
                <artifactId>ollama4j</artifactId>
                <version>1.1.6</version>
        </dependency>
</dependencies>
```

Integrating Ollama and SARL



Integrating Ollama and SARL



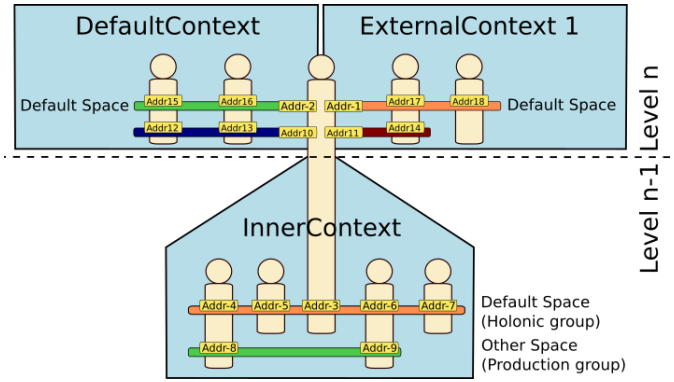
Well Done!!

Photo by Elias Maurer on Unsplash

SARL has more to offer

Holonic Agents

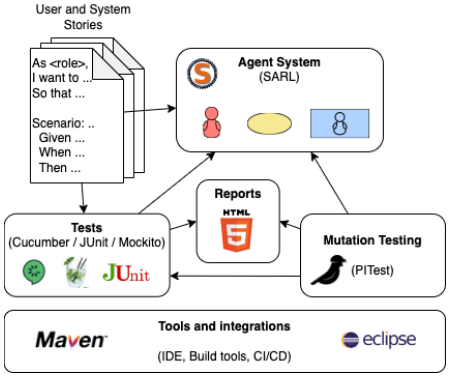
- Compose agents recursively with ease
- Integrate agent subsystems
- Implement complex system

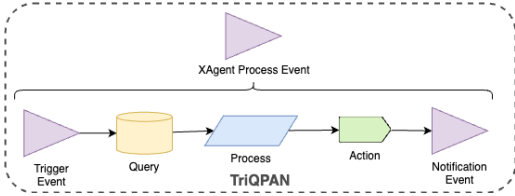


Agent Architectures

- Goal reasoning agents
- Explainable Agent (XAg)

Simulator Architecture





SARL DevOps support

- Model Software Engineering practices support
- Test first approaches
- Error Handling in Agents

Survey

https://forms.gle/5Q8rB5AnYH1Wmcg9A

SARL

Agent Programming Language

```
agent Sebastian {
  uses DefaultContextInteractions
  on Finished {
    emit(new Greeting => [
        message = "Thank you!"
    ])
    }
}
```

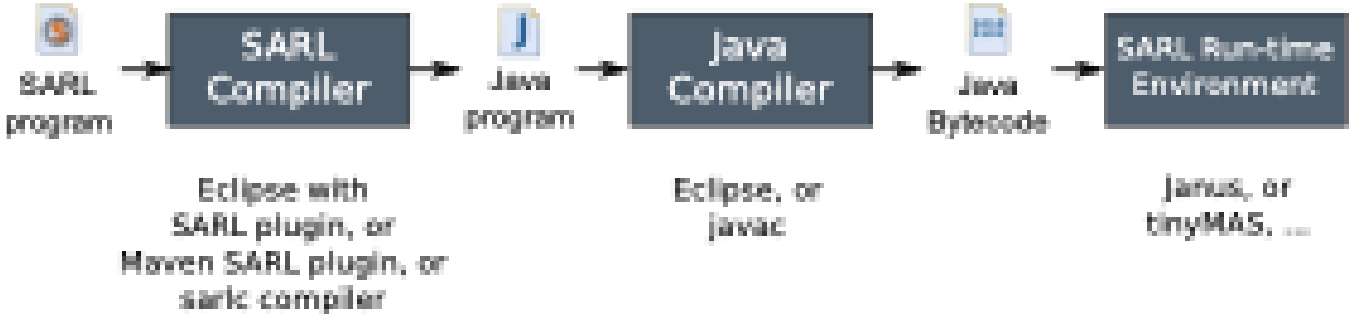
sebastianrodriguez

Sebastian Rodriguez - sebastian.rodriguez@rmit.edu.au

RMIT UNIVERSITY Ready for what's next

Summer Workshops 2026 – S&R Hub - 27 Feb 2026

SARL Compilation process



SARL is 100% compatible with JAVA