Introduction to SARL programming

-SUNIL KARAMCHANDANI

Table of content

Abstract

- Java as a language, although widely used, is relatively slow and most importantly environment independent.
- This restricts the flexibility of usage of Java to a wide array of problems which can be overcome by using SARL.
- This paper serves as a tutorial for beginners who wish to exploit the use of SARL as an agent oriented programming.
- The paper does not debate agent over object however it employs the use of object oriented programming as a thread for the agents.
- When agents operate independently in a program, SARL reduces to the Java platform.

Introduction

- With advancements in the IoT industry and increasing demand for intuitive intelligent languages, SARL, a general purpose agent oriented programming language has shown great promise in developing software with ease.
- It can be very well used for mobility/ transport systems analysis and simulation. As certain environmental changes need to occur for the agents to get stimulated, it can be used to implement secure websites; and hence expand its applications in the field of Cyber Security and secured telecommunications.
- On reading this paper, one would easily be able to understand the basics of SARL programming. However, an in-depth knowledge on SARL can be obtained by learning the basics from “http://www.sarl.io/docs/official/index.html” following which more information can be obtained from https://github.com/sarl/sarl.

What is SARL programming?

- It is a modular agent-oriented programming language.
- Its intuitive environment gives freedom for creating the agents and the inter-agent communication.
- Additionally, SARL applications are able to use any external Java library. It uses the Janus run-time environment. It is an open-source Java 1.8 platform that enables developers to quickly create web-based, enterprise and desktop applications. It provides a wide range of features for developing, running, viewing, and monitoring multi-agent applications.
- SARL codes are converted to JVM bytecode and hence we can call java methods with ease using sarl agents. Also, it is open sourced under Apache licensed version 2.0.
- An important feature of the SARL programming language is its native support for "holonic multiagent systems," and "recursive agents“

What is SARL programming?

- Agents are self-driven entities which work towards achieving a particular goal and respond to changes in an environment.
- Agents have four important features:

i. Autonomy -They have control over their own actions and use adaptation learning

ii. Reactivity -They can react in a timely manner to event changes, hence can be used for real-time operations

iii. Pro-activity -They work towards a specific goal; identifying opportunities and taking initiatives

iv. Sociability -Multiple agents can interact with each other using a specific language.

- These agents are composed of other agents, giving new abilities to the parent.

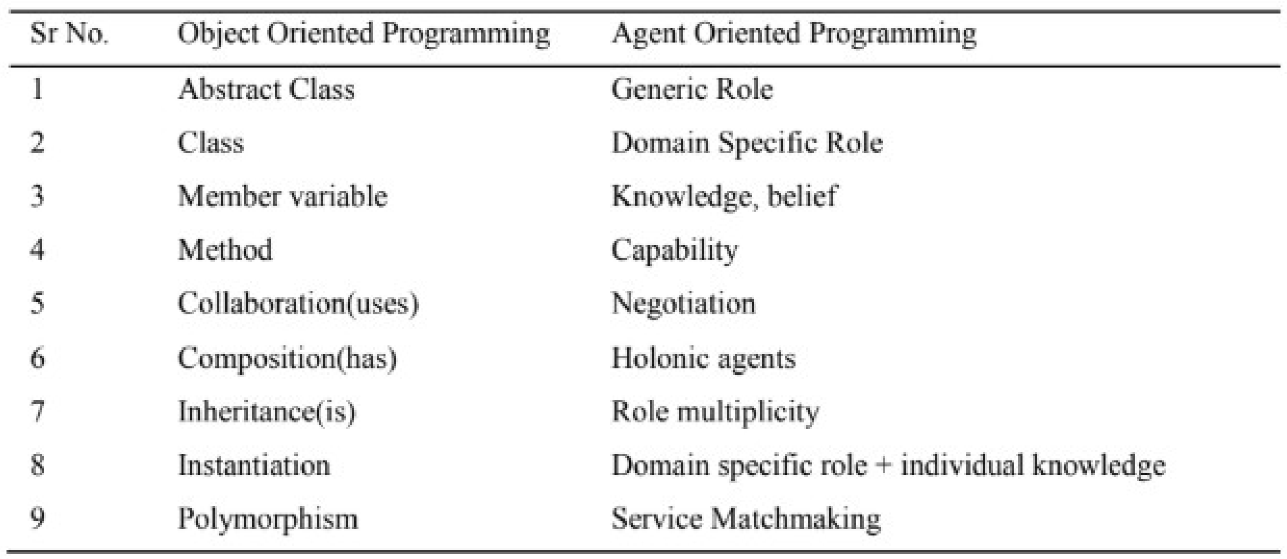
What is SARL programming?

- The environment, as a space shared between agents, is a key component of multiagent systems. Depending on systems, this space may integrate physical, communication or social dimensions where agents interact.
- Each dimension of the environment has its own processes and rules to support its interaction model.
- Multiagent-based simulation model is composed of three parts namely behaviour of agents, the interactions amongst agents and the definition of the environment.

What is SARL programming?

- Apart from this, SARL defines reusable components to describe agent behavior- capacity and skill.
- Capacity defines a collection of actions while skill defines the implementation of capacity. The keyword “Uses ” is used to extend an agent.

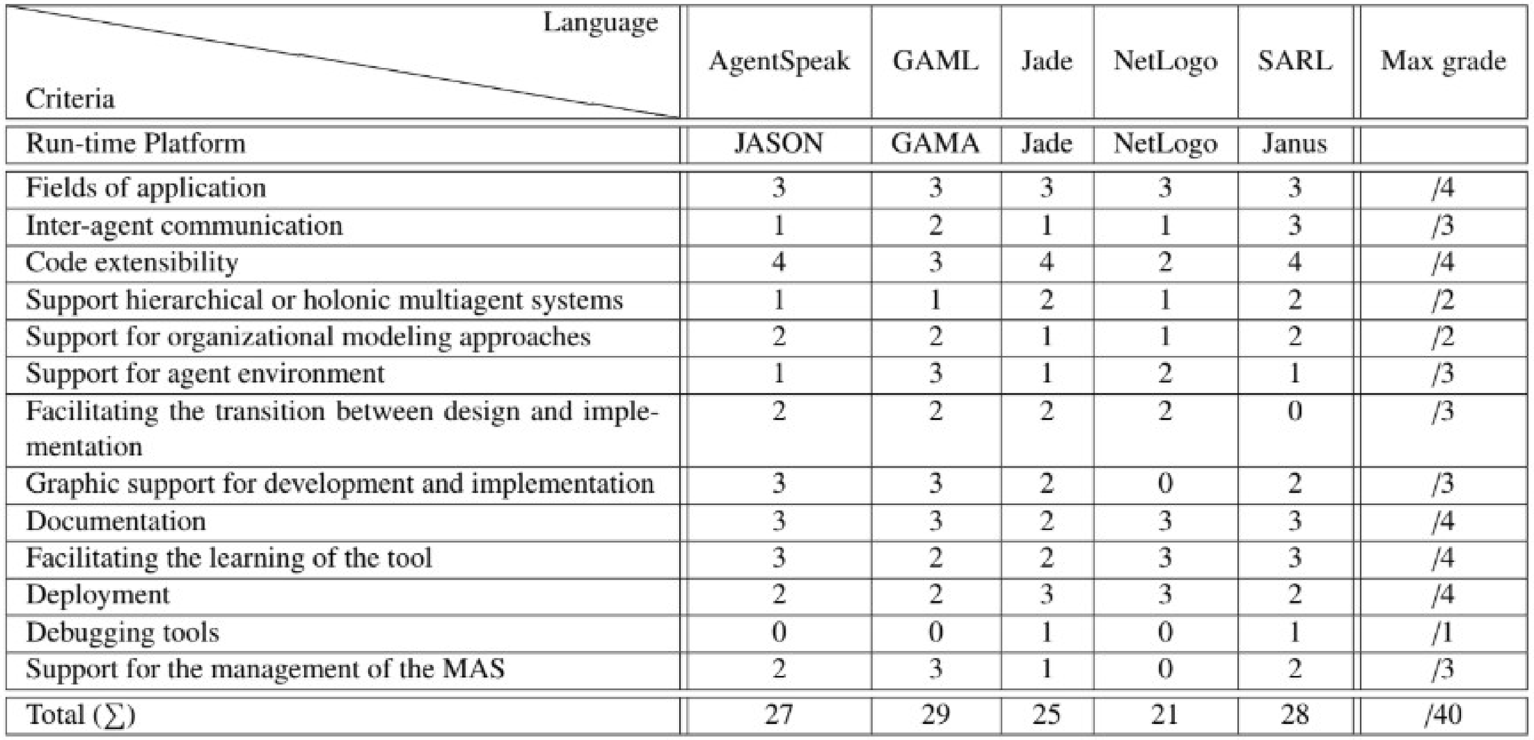
Difference between Object and Agent oriented programming



Sr No.	Object Oriented Programming	Agent Oriented Programming
1	Abstract Class	Generic Role
2	Class	Domain Specific Role
3	Member variable	Knowledge, belief
4	Method	Capability
5	Collaboration(uses)	Negotiation
6	Composition(has)	Holonic agents
7	Inheritance(is)	Role multiplicity
8	Instantiation	Domain specific role + individual knowledge
9	Polymorphism	Service Matchmaking

Difference between SARL and other agent oriented languages:

- In the paper titled “First Comparison of SARL to Other Agent-Programming Languages and Frameworks”, the authors have compared SARL to other agent oriented programming languages like AgentSpeak, GAML, Jade, NetLogo and MaxGrade, and we see that SARL offers a true complete dedicated and developed solution for multi-agent systems.



Criteria \ Language	AgentSpeak	GAML	Jade	NetLogo	SARL	Max grade
Run-time Platform	JASON	GAMA	Jade	NetLogo	Janus	
Fields of application	3	3	3	3	3	/4
Inter-agent communication	1	2	1	1	3	/3
Code extensibility	4	3	4	2	4	/4
Support hierarchical or holonic multiagent systems	1	1	2	1	2	/2
Support for organizational modeling approaches	2	2	1	1	2	/2
Support for agent environment	1	3	1	2	1	/3
Facilitating the transition between design and implementation	2	2	2	2	0	/3
Graphic support for development and implementation	3	3	2	0	2	/3
Documentation	3	3	2	3	3	/4
Facilitating the learning of the tool	3	2	2	3	3	/4
Deployment	2	2	3	3	2	/4
Debugging tools	0	0	1	0	1	/1
Support for the management of the MAS	2	3	1	0	2	/3
Total (∑)	27	29	25	21	28	/40

Difference between SARL and other agent oriented languages:

- From this table, we see that SARL is better or comparable to other agent oriented programming languages in terms of fields of application, inter-agent communication, code extensibility, holonic multiagent systems, support for organizational modelling approaches, documentation and debugging .

Getting started with SARL programming-Installation

- Installing SARL is very straight forward.
- To install SARL on any of the operating systems all we need to do is to go to the official SARL website (http://www.sarl.io).
- On the home page there are two options in the start Download and Documentation. By clicking on Download it takes you to another webpage where we need to select the operating system we want it to install it on. It has all the major operating system option available there like windows (32 & 64 bit), mac os and linux.
- By clicking on the operating system of your choice you will download the setup files compatible with your operating system in an archive file format which can then be unzipped on to the location of your choice.
- The website has the latest and stable version for download always.

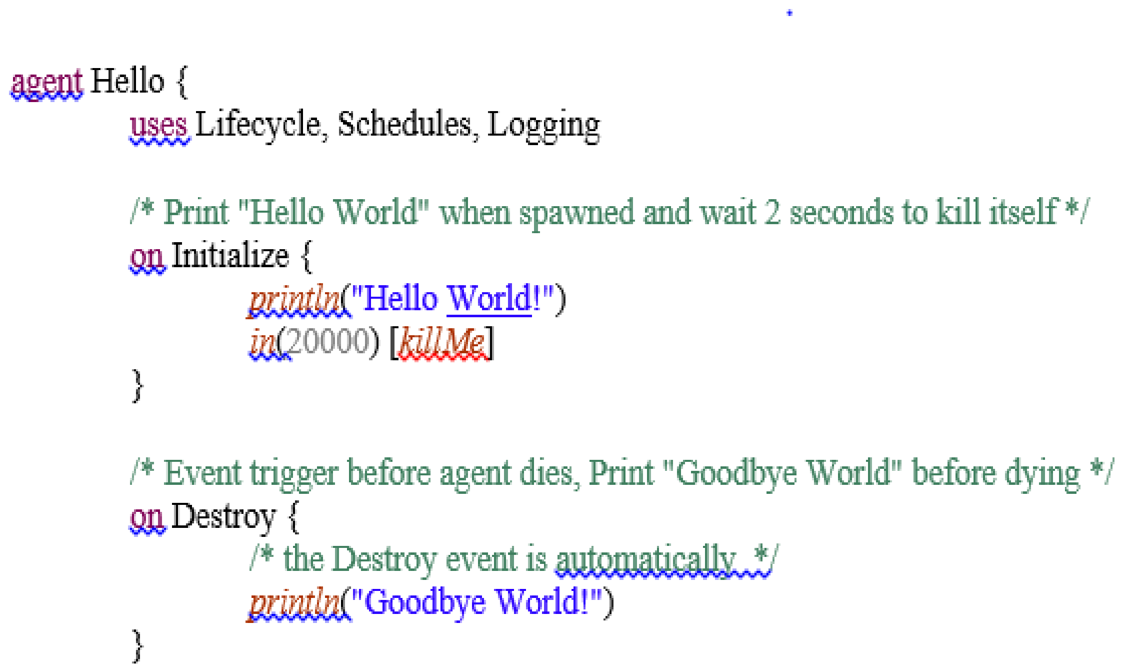
Getting started with SARL programming- Installation

- One thing to be kept in mind while downloading SARL is that Eclipse works on Java Bytecodes; hence it requires Java run-time environment to be installed.
- This can be easily installed from http://www.oracle.com/technetwork/java/index.html.
- Ensure that you download Java run-time environment version 1.8 or higher. After the zip file has been downloaded, unzip it close to the root; preferably C folder as the zip contains deeply nested folder structure. This zip file contains all the software needed for installation.
- It has applications like JANUS which are solely based for SARL language and even applications like ECLIPSE which are used for java but are also compatible with SARL.
- The extracted files also have all the required plugins, configuration files and workspace in it.

Example codes-Hello World

- To begin with SARL programming, let us consider the code to print “Hello World” (also present in basic codes).The steps involved are as follows:

1. Define an agent named Hello using the “agent” keyword.
2. Define the events using “uses” keyword.
3. Initialize and Destroy are Lifecycle events. The event Initialize is used to begin execution while the event Destroy is used to stop execution when some condition is met.
4. The condition in this case is a 20000ms schedule. After 20000ms, the event ends as “killMe” is executed and moves onto the next event. The syntax to define this schedule is in(20000) [killMe].
5. Logging capacity permits the usage of println.
6. Hello World is printed when the event is initialized using println.
7. Similarly Goodbye World is printed when the event is destroyed.
8. This code can be executed by clicking on run✉run-as✉sarl agent.



```
agent Hello {
        uses Lifecycle, Schedules, Logging

        /* Print "Hello World" when spawned and wait 2 seconds to kill itself */
        on Initialize {
                println("Hello World!")
                in(20000) [killMe]
        }

        /* Event trigger before agent dies, Print "Goodbye World" before dying */
        on Destroy {
                /* the Destroy event is automatically  */
                println("Goodbye World!")
        }
```

Example codes-Countdown

- Another code which helps us understand coding in SARL is implementing Countdown.
- The following code can be used as a timed alarm and uses in-built functions like AtomicInteger and DecrementAndGet.
- The following code is also available amongst the basic codes. In this case, it is slightly modified to act like an alarm.

Example codes-Countdown

- The explanation of the code is as follows:

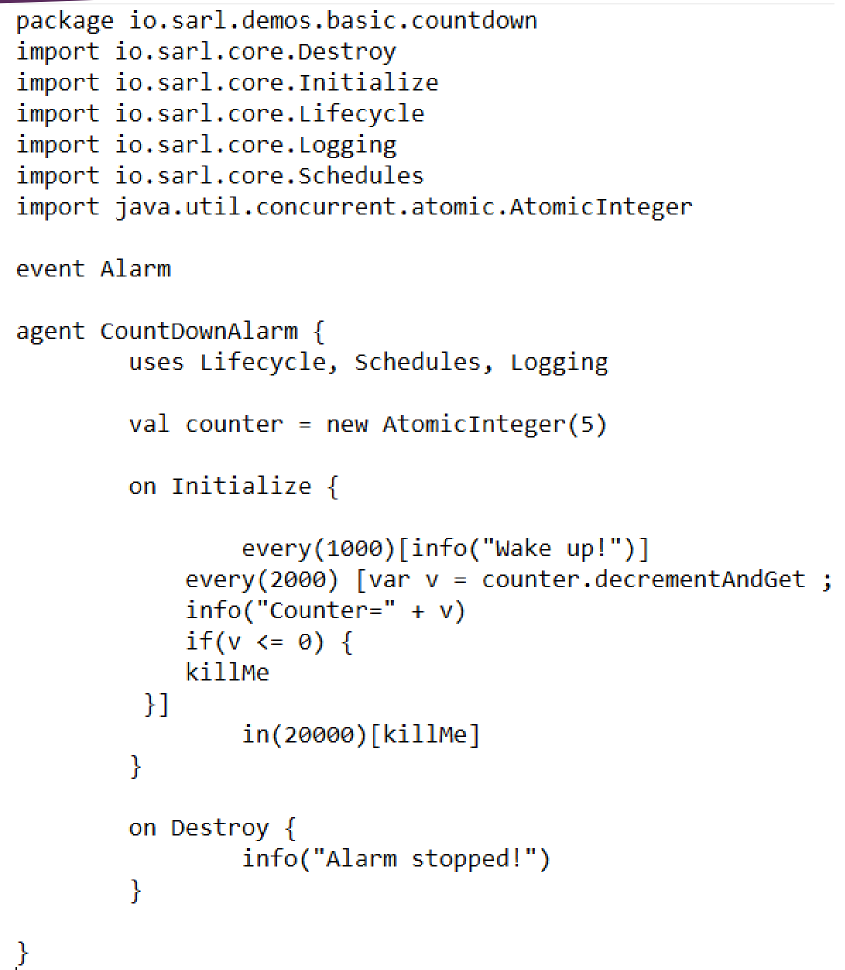
• Import the required events from io.sarl.core. The new event specific to this code is the AtomicInteger extracted from java.util. This event is used to set up a variable which can be updated i.e. incremented or decremented.

• Define an event Alarm using “event” keyword

• Define an agent CoutDownAlarm and setup required events using “uses” keyword

• Setup a counter to an arbitrary value using AtomicInteger from which it will get decremented to zero

• Decrement counter and print “wake up” as many times as needed. After the schedule is over, display “Alarm stopped!” to indicate end of execution.



```
package io.sarl.demos.basic.countdown
import io.sarl.core.Destroy
import io.sarl.core.Initialize
import io.sarl.core.Lifecycle
import io.sarl.core.Logging
import io.sarl.core.Schedules
import java.util.concurrent.atomic.AtomicInteger

event Alarm

agent CountDownAlarm {
        uses Lifecycle, Schedules, Logging

        val counter = new AtomicInteger(5)

        on Initialize {

                every(1000)[info("Wake up!")]
            every(2000) [var v = counter.decrementAndGet ;
            info("Counter=" + v)
            if(v <= 0) {
            killMe
     }]
                in(20000)[killMe]
        }

        on Destroy {
                info("Alarm stopped!")
        }

}
```

Example codes-Implement logic gates

- After understanding the example codes, some modifications to understand SARL better should be implemented on the example code. In this code, we use inbuilt functions(countdown) to implement logic gates.
- After importing the same packages as that in the Countdown code, the following changes need to be done in the countdown code for implementing logic gates:

Example codes-Implement logic gates

```
agent CountDownAgent {
        uses Lifecycle, Schedules, Logging
        val counter = new AtomicInteger(1)
        val v=counter;
        val counter1 = new AtomicInteger(1)
        var w = counter1;
        val count = new AtomicInteger(4)
        on Initialize {
                info("Initialize")
                every(2000) [ /* Decrement counter every 2 sec until v = 0, then kill itself*/
                        if (v == 1 && w==1) {
                                info("output=0")
                                var w = counter1.decrementAndGet;
                                var count = count.decrementAndGet;
                                if(count==0){
                                        killMe
                                }
                        }
                        else if(v== 1 && w==0) {
                                info("output=1")
                                var v = counter1.decrementAndGet;
                                var w = counter1.incrementAndGet;
                                var count = count.decrementAndGet;
                                if (count == 0) {
                                        killMe
                                }
                                }
```

```
                        else if (v == 0 && w == 1) {
                                info("output=1")
                                var w = counter1.decrementAndGet;
                                var count = count.decrementAndGet;
                                if (count == 0) {
                                        killMe
                                }
                        }
                        else if (v ==0 && w==0) {
                                info("output=0")
                                var count = count.decrementAndGet;
                                if(count==0){
                                        killMe
                                }
                                }
                }
                on Destroy {
                        info("This is the required truth table")
                }
        }
```

Example codes-Implement logic gates

- The explanation of the code is as follows:

• Update the previous (countdown) code by making a few changes. Define v and w as two variables using AtomicInteger and set their values as 1 each.

• Increment and decrement these variables using decrementAndGet and incrementAndGet.

• Print values depending on required truth table.

• The structure of if-else if loop is explained here. A detailed explanation on different loops is provided in sarl documentation.

- Similar codes can be written to understand the basics of SARL better.

Conclusion

- With a good understanding of the working of agents and how they react to changes in the environment, SARL can be used for a large number of tasks.
- The language used is intuitive and simple to understand.
- Although SARL is one of the best agent oriented programming languages, it can be improved further by bridging the gap between design and implementation, improving deployment features and implementation tools.

References

[1] Stéphane Galland, Sebastian Rodriguez, Nicolas Gaud(2017) “Run-time environment for the SARL agent-programming language: the example of the Janus platform” *Future Generation Computer Systems*.

[2] Sebastian RODRIGUEZ, Stephane GALLAND, Nicolas GAUD (2015) “A New Perspective on Multi-Agent Environment with SARL.” *Procedia Computer Science* 56 (2015) 526 – 531.

[3] Maxime Feraud, Stephane GALLAND (2017) “First Comparison of SARL to Other Agent-Programming Languages and Frameworks.” *Procedia Computer Science 109 (2017) 1080–1085.*

[4] S. Galland, N. Gaud, S. Rodriguez, V. Hilaire, “Janus: another yet general-purpose multiagent platform” *The 7th Agent-Oriented Software Engineering Technical Forum (TFGAOSE-10), Agent Technical Fora, Agent Technical Fora, Paris, France, 2010*.

[5] O. Zohreh Akbari (2010) “A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance” *Journal of Computer Engineering Research Vol. 1(2), pp. 14 - 28, April 2010*

[6] Bergenti, F, Gleizes, M-P, Zambonelli, F (2004) “Methodologies and Software Engineering for Agent System” *The Agent Oriented Software Engineering Handbook. New York: Kluwer Academic Publisher*.

[7] Sebastian Rodriguez, Nicolas Gaud, Vincent Hilaire, Stephane Galland, and Abderrafiaa Koukam(2006) “An analysis and designconcept for self-organization in holonic multi-agent systems” *In the International Workshop on Engineering Self-OrganizingApplications (ESOA’06), pages 62–75. Springer-Verlag, May2006.*

[8] Rafael H. Bordini, Mehdi Dastani, Jurgen Dix, Amal El Fal-lah Seghrouchni(2009) “Multi-Agent Programming: Languages, Toolsand Applications” *Springer Publishing Company, Incorporated,1st edition, 2009*

Thank You!