Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2016) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The International Workshop on Agent-based Modeling and Applications with SARL (SARL 2017)

First Comparison of SARL to Other Agent-Programming Languages and Frameworks

Maxime Feraud[b], Stéphane Galland[a,b,*]

[a] *LE2I, Univ. Bourgogne Franche-Comté, UTBM, F-90010 Belfort, France*
[b] *Computer Engineering Dept., Université de Technologie de Belfort-Montbéliard, F-90010 Belfort, France*

Abstract

This paper proposes a first comparison of different agent-oriented programming languages, including the SARL agent-programming language. The study of those tools is based upon various criteria that are defining the essential aspects of multi-agent systems. For the sake of clarity, the comparative study is presented in a tabular form, including the languages and all the criteria. For each criterion, a definition is made to understand its meaning. At the end of the article, all the languages will be described in order to define their advantages and disadvantages.

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Multi-Agent System, Programming Language, Language Comparison, SARL

1. Introduction

During the last years, multiagent systems (MAS) have taken their place in our society. Application fields include robotics, artificial inelligence, cinema, video games. This evolution is the answer to increasingly complex projects which require "intelligent" systems. Multiagent systems allow to implement solutions with intelligence, capable of reasoning, learning and interacting between different agents.

These systems represent a totally different way of looking at things. This way of designing systems resulted in new tools, methodologies and architectures, better suited to MAS modeling, e.g. ASPECS[1], MaSE[2] or even Gaia[3]. In addition to these tools, the main programming languages have developed platforms for agent oriented programming. There are several dozens, e.g. JADE, NetLogo, SWARM, MATSIM or GAMA.

These solutions are highly interesting because they frame and provide a methodology for the development of agent-based systems. On the other hand, these systems are very complex to implement, and the conventional programming languages are not suited. There is therefore a real need for programming languages dedicated to MAS, which would offer more clarity to developers, and simplify developments. In this paper, a comparative study of agent oriented programming languages, including the SARL agent programming language is presented. *This work was done in the*

* Corresponding Author.
E-mail address: stephane.galland@utbm.fr (Stéphane Galland).

Peer-review under responsibility of the Conference Program Chairs.

context of the MAS introduction lecture of UTBM[1]. The SARL language tries to set up adaptive and modular principles for developing multiagent systems. The goal of this paper is to highlight the advantages and disavantages of SARL against the other frameworks in order to build the development milestones for SARL.

This paper is structured as follow. Section 2 explains the comparison criteria, and their uses. Section 3 presents the result of the comparison in a synthetic table. Section 4, the compared languages are discussed, in order to define the positioning of the language SARL against the other studied languages and frameworks. Finally, the paper is concluded, and perspectives of this study are provided.

2. Comparison Criteria

For the study of the different languages and frameworks, comparison criteria are defined and selected. They are inspired by those provided by Garneau and Delisle[4]. In order to make the evaluation process easier, and clear the evaluation results, each criterion is associated to a number scale, going from 0 to 4. By the way, a value equal to zero means that the criterion is not relevant. And, a value equal to 4 means that the criterion is fully relevant. In the rest of this section, the descriptions, which are explaining the exact meaning of the scale values are provided for each criterion.

2.1. Description of comparison criteria

- **Fields of application:** MAS have a very wide scope of application, including 3D simulation, geography simulation, social simulation, video game... For each field of application, the language is defined according to a specific ontology. The score depends on the genericity of the language:
  1 - No scope of application
  2 - One domain
  3 - Multiple domains
  4 - Generic or application-independent

- **Inter-agent communication:** In the source code, the language enables to set-up communications between several agents. How is this supported by the language? Is this management simply or necessary for development?
  0 - No communication (single agent platform)
  1 - agent-to-agent communication (message-based or event-based)
  2 - agent-to-agent communication and stigmergy (communication through the environment)
  3 - Language allows everything (events, messages, no constraint on the mode of interaction)

- **Code modularity:** This is the ability for the code to be easily adapted to a change. It includes the language ability to provide mechanisms and properties for maintaining the scalability of the system. It includes also the ability of the language to redefine, refine, reifine existing code or modules.
  1 - No modularity
  2 - Static modularity (definition and use of modules in the language)
  3 - Dynamic loading of modules (creation and use at any time during the run-time)
  4 - Using external libraries, which may be written in other programming languages

- **Support hierarchical or holonic multiagent systems:** Several languages provide the support for defining and deploying hierarchies of agents (agents composed of agents). This criterion classifies the ability of a language or a framework to create agent hierarchies, and specify the relationship between the agents at the different levels. This is a key criterion for SARL and the associated Janus platform.
  0 - No hierarchy support
  1 - Static definition of the hierarchies
  2 - Dynamic (run-time) creation of agent hierarchies

[1] UTBM: Université de Technologie de Belfort-Montbéliard

- **Support for organizational modeling approaches:** MAS are usually complex systems according to the definition of Odell[5]. Designing a complex system may be done with an organizational approach, based on the concepts of organization, groups and roles for instances[1]. MAS programming can be based on the same metamodel as the one used durnig the system design. This criterion focuses on the existence of these metamodels. This is a key criterion for SARL and the associated Janus platform.
  0 - No organizational modeling (no specific statement or concept in the metamodel)
  1 - Static organizational modeling
  2 - Dynamic organizational modeling

- **Support for agent environments:** This criterion focuses on the ability of the language or framework to support agent environments, as defined by Weyns et al.[6]. Does the language provides the statements to define the elements and the structure of the agent environment. Are there language statements to access to the agent environment's components?
  0 - No agent environment
  3 - No constraint on the type of agent environment, but no environment template provided
  1 - Discrete environment (matrix) is provided
  2 - Non-discrete environment (geographical information system, 3D space...) provided

- **Facilitating the transition between design and implementation:** Is the language made the implementation easy from the designed model? MAS projects being complex, a simplified code implementation avoid a great loss of time. Do the related tools allow (semi-)automatic implementation?
  0 - Very complicated
  1 - Complicated
  2 - Simple
  3 - Very simple

- **Graphic support for development and implementation:** Is there an integrated development environment (IDE), or a development and implementation platform that offers graphic support simplifying the implementation of the projects?
  0 - No IDE (just a program compiler)
  1 - Just an editor with syntax coloring, and a dedicated compiler
  2 - Editor with syntax coloring and helping features, such as auto-completion
  3 - Editor with helping features, creation wizards and automatic code generation

- **Documentation:** This criterion defines the quality of the documentation around the language or the framework. Are there documentations on the internet? Are they easily accessible? Is the API associated to the language or the framework well detailed?
  1 - No documentation
  2 - API documentation only
  3 - API documentation, and reference/standard document(s)
  4 - API, standard/reference document(s) and tutorial(s)

- **Facilitating the learning of the tool:** In relation with the documentation criterion, this criterion emphasizes the simplicity of use for the language or the framework. Is the syntax intuitive? Is the syntax close to an known languages? Is it a machine-oriented, object-oriented, agent-oriented language? Students have provided indivudual subjective answers to this criterion. If a student has not realised the expected tasks with the framework during the lab work, he must provide the score 1 or 2. All aspects that will make the handling of a simple language:
  1 - Very complicated
  2 - Complicated
  3 - Simple
  4 - Very simple

- **Deployment:** Issues related to the deployment of the agents on one or more computers constitute a key problem in MAS domain. Does the language or the framework provide features or tools for making easier the

agent deployment? Do an external tool needed for the specification and the execution of the deployment policy? This criterion relates each language to their run-time platform, if they exist one:
1 - No deployment tools
2 - Local computer deployment tool
3 - Static Computer network deployment tool
4 - Dynamic/Automatic network deployment tool

- **Debugging tools:** Is there a tool for debugging the MAS, greatly facilitating the development of the projects?
  0 - No
  1 - Yes

- **Support for the management of the MAS:** Once the development is done, and if the language has a support, is it possible to manage the MAS (agent creation/deletion, resource management...) If so, is it simple or not?
  0 - No MAS management support
  1 - Static definition of the MAS
  2 - Run-time agent creation and removal tools
  3 - Exploration tool (according to the FIPA platform features)

3. Comparison Table

Table 1 shows the comparison results among the studied languages and platform. We have selected SARL, AgentSpeak, GAML, Jade, and Netlogo as the more relevant in the context of software engineering lecture on MAS. Based on the 14 criteria, a final score is assigned to each language or framework. Higher is the score, better it is responding to our evaluation criteria. The evaluation is based on the average individual scores, which are provided by 24 students of UTBM, who are learning MAS engineering and implementation. Each framework was used by each student during a 1-hour laboratory work session.

Table 1: Scores to each comparison criterion for the studied agent programming languages or frameworks

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 ($\sum$)	27	29	25	21	28	/40

4. Discussion

In the previous section, the evaluation results regarding the comparison of agent programming languages and frameworks are presented. In this section, the results are synthetized and discussed in order to highlight the different advantages and disadvantages of each language or framework. Finally, this section is concluded by a discussion on the improvements to SARL to be made compared to the other languages and frameworks.

AgentSpeak/Jason[7] is a declarative language. AgentSpeak allows to develop a set of plans that agents will follow according to different situations. Jason is a Java-based framework that is supporting an extended version of AgentSpeak. It offers the means to implement complex multiagent systems thanks to its extensibility. It combines agent environments and agent architecture to create customized agent combinations. This allows the developer to create software agents that are simulating realistic scenarios. Customizing the agent's architecture enables to implement a local representation of the agent environment. In this environment, it is possible to carry internal actions out that are exploiting the local representation of the environment. This approach offers the possibility to model and implement a large number of agent behavior in relation to the environment. One of the disadvantages of AgentSpeak, in relation with its declarative and logic-based nature is that its use is not very intuitive, compared to other programming languages. However, using such language provides the ability to execute agents directly to Jason, making it a fairly versatile system.

GAML/GAMA[8] is an environment for development, modeling and simulation of spatial systems with MAS. This framework provides a complete environment, which is ideal for building spatial or geographical simulation. Its main advantages rely on its great versatility, e.g. application domain independency, and the simplicity with which it is possible to define a model. With this framework, it is equiped with a rich and accessible modeling language, named GAML. Based on XML, it allows to define complex models, which are integrating both geographical vector data and entities at different scales. Most agent-based simulation platforms only allow the use of a grid-based environment. Frameworks, which support geographical data are more rare. Additionally, we think these platforms are often very complex to use, and require an effort of understanding, which represents a heavy investment in time. GAMA was developed to address these issues. It allows geographical data to be seamlessly identified and integrated into the agent-based simulation. The agentification facilitates the integration of dynamic behaviors in the simulation.

Jade[9] is a very popular agent platform. It simplifies the implementation of multiagent systems through a middleware that complies with the FIPA specifications, and through a set of tools that support the debugging and deployment phase. Each agent is composed of different and competing behaviors. Agents can be added dynamically in the multiagent system. Jade includes a run-time environment in which Jade agents can "live". Among the advantages that Jade offers, there is a wide variety of provided graphical tools for the management and monitoring of the agents and exchanged messages during the run-time. In addition, Jade can be integrated and deployed easily with different software via an extensive API. On the other hand, one of the biggest disadvantages of Jade is that mobility and embedded systems are not a key elements in Jade. Emphasis was placed on other features, which target the development of "general" multiagent systems. The second key disavantage is the complexity of the programming API that makes harder to develop an application with Jade.

Netlogo[10] offers a simulation software based on MAS. It has its own, specific, easy to use programming language (LOGO), as well as high-level structures and built-in functions for the common tasks for the agent behaviors. It is one of the most popular platforms, especially among the academic community. Netlogo suffers in spite of himself from an a priori, namely that it is not a serious research tool. Usually, MAS developers consider that it should only be used as a teaching tool, or as a platform for the rapid prototyping of MAS. This poor image of Netlogo is due to an extended set of predefined models, and the related grapihcal user interface (GUI) elements, that put the emphasis on pedagogical applications. However, Netlogo works perfectly for building MAS. The GUI elements simplifies the development of the MAS models that are sometimes complex to implement in other languages and frameworks. In addition, it offers the ability to run models headless in order to improve model execution performances. The simplicity of Netlogo enables to reduce considerably the programming skills that are required to develop and simulate a MAS. Specific models inside the Netlogo environment, e.g mobile agents acting in a two-dimensional grid may be used as templates for building applications. Finally, it also contains many sophisticated features: agent behavior templates, agent lists, predefined GUI...

SARL[11] is a language intended to provide the fundamental abstractions to deal with several MAS aspects: distribution, interactions, decentralization, reactivity, autonomy and dynamic reconfiguration. Authors of SARL consider that these features represent the main requirements for the implementation of complex and modern software MAS applications, all in the simplest way as possible. SARL is developed on top of Xtext and Xbase, and inherits key features such as extension methods, closures and many more. The inherited features constitute a force for SARL that enables developers to create agents very quickly thanks to an intuitive syntax. In addition, SARL defines specific keywords for MAS development, and reusable components to describe the agent's behaviors: the concepts of capacity and skill. Agent, capacity, skill and behavior concepts are easy to extend. The SARL agents are actually holons. They are agents composed of other agents, which can provide new abilities to their parents. SARL can be used with a set of tools, platforms or framework to support its execution, such as the Janus platform or the TinyMAS platform. Unfortunately, SARL does not have programming tools for specifying the deployment of the MAS. Janus is the official run-time environment for SARL. 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 multiagent applications. Janus applications can be distributed in a network with ab "zero configuration" approach, i.e. Janus nodes are dynamically discovered across the local computer network. Janus also manages natively the concept of holon.

5. Conclusion and Perspectives

In this paper, a first comparison of the SARL agent-programming language and other languages and frameworks is presented. This work was done in the context of the MAS introduction lecture at the UTBM. Key criteria are presented, and scores are given to each framework. For concluding this study, we note that no language is perfect. SARL seems to offer a more complete response to the problems of agent-oriented programming. However, some aspects can still be improved, e.g. implementation tools and wizards, agent environment support, GUI support, deployment features. Netlogo and GAMA provide a nice set of features that could serve as source of inspiration for extending SARL/Janus. However, SARL offers a true complete dedicated and developed solution for MAS. We note the freedom for defining the agent environment, the inter-agent communication. Additionally, SARL applications are able to use any external Java library.

This study will be extended with more criteria (community liveliness, run-time features...) Existing and new criteria will be formally defined, and put in a more complete document in order to clearly define the key features to put in the roadmap of SARL. Additionally, more frameworks will be added in this comparison in the future.

References

1. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.. 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* 2010;**2**(2):260–304.
2. DeLoach, S.A.. *The MaSE Methodology*. Boston, MA: Springer US; 2004, p. 107–125.
3. Wooldridge, M., Jennings, N.R., Kinny, D.. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems* 2000;**3**(3):285–312.
4. Garneau, T., Delisle, S.. Programmation orientée-agent : évaluation comparative d'outils et environnements. In: *Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multiagents*. Lille, France; 2002, p. 1–13.
5. Odell, J.. Agents and complex systems. *Journal of Object Technology* 2002;**1**(2):35–45.
6. Weyns, D., Omicini, A., Odell, J.. Environment as a first-class abstraction in multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 2007;**14**(1):5–30.
7. Píbil, R., Novák, P., Brom, C., Gemrot, J.. *Notes on Pragmatic Agent-Programming with Jason*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-31915-0; 2012, p. 58–73.
8. Drogoul, A., Amouroux, E., Caillou, P., Gaudou, B., Grignard, A., Marilleau, N., et al. GAMA: multi-level and complex environment for agent-based models and simulations (demonstration). In: *international conference on Autonomous agents and multi-agent systems*. United States; 2013, p. 1361–1362.
9. Bergenti, F., Caire, G., Gotta, D.. Agents on the move: Jade for android devices. In: Santoro, C., Bergenti, F., editors. *WOA*; vol. 1260 of *CEUR Workshop Proceedings*. CEUR-WS.org; 2014, .
10. Tisue, S., Wilensky, U.. Netlogo: Design and implementation of a multi-agent modeling environment. In: *Proceedings of the Agent2004 Conference*. 2004, .
11. Rodriguez, S., Gaud, N., Galland, S.. Sarl: A general-purpose agent-oriented programming language. In: *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*; vol. 3. 2014, p. 103–110.