Fantastic MASs and where to find them: First results and lesson learned

Daniela Briola[1][0000−0003−1994−8929], Angelo Ferrando[2][0000−0002−8711−4670], and Viviana Mascardi[2][0000−0002−2261−9926]

[1] University of Milano-Bicocca, Milano, Italy
`daniela.briola@unimib.it`
[2] University of Genova, Genova, Italy
`forename.surename@unige.it`

Abstract. Nowadays, the Multiagent Systems research community is facing new challenges related to engineering the overall process of software development, tailoring it to the specific needs of the community, and integrating SE techniques into many studies in the MAS area. More and more frequently, researchers need already developed MASs for validating their new proposals. Often, they spend time in looking for the code of existing tools to compare with the state of the art. Unfortunately, accessing this kind of resources, which are the starting point for many SE activities, is not always easy. In this paper, we present the first outcome of the initiative "Fantastic MAS and where to find them", launched in June 2022, where we asked the agent community to contribute in the creation of a repository to facilitate the sharing of the already existing tools (agent development frameworks, libraries, add ons of already existing platforms) and MASs with their code. The "Fantastic MAS" goals are to i) improve the sharing and reusing of research results ii) support the SE activities in our research community, iii) help making a step towards the Open Science movement, which has been already widely adopted in other research communities. Besides providing an overview of the submissions we got, we discuss the open problems that emerged in these eight months of the initiative, so that to stimulate the discussion in the community.

Keywords: Agent-Oriented Software Engineering, Multiagent systems, Agent development framework

1 Introduction

In the last years, the research community related to agents and Multiagent Systems (MASs) opened more and more to software engineering problems related to all the aspects of designing and implementing MASs [39]. This is mandatory to face the growing complexity of this variegate research field: indeed, testing approaches and tools, formal methods, simulators of MASs, and design methodologies are emerging in our community, in response to the need of researchers and in line with the advances in the Software Engineering (SE) research area.

As a community, we are consequently facing the problems associated with this transformation toward a more "software engineering approach", which provides great benefits but also requires a change in the way the research results are presented and shared. The Software Engineering community has adopted from way back a systematic approach regarding how research results should be presented, above all in the top level scientific venues, which follows a sort of standard and a common list of requirements to assure the quality of the published products and to enhance their sharing and reusability. The underlying idea is that the value of a research result relies not only in the new proposal presented in the paper itself, but in the associated produced tool, exploited data, and whatever was used for achieving the results presented in the paper. So, the research result has a double value, the state of the art contribution and the availability of associated data and software to be reused by the community. This approach is in line with the view proposed by the Open Science movement[3], which is getting more and more attention by all the academic world. Considering the top rated SE conferences (ICSE[4], ESEC/FSE[5], ICST[6]), their call for papers clearly mention the *"Open Science Policy"*, stating that:

"the steering principle is that all research results should be accessible to the public and, if possible, empirical studies should be reproducible. In particular, we actively support the adoption of open data and open source principles and encourage all contributing authors to disclose (anonymised and curated) data to increase reproducibility and replicability. Note that sharing research data is not mandatory for submission or acceptance. However, sharing is expected to be the default, and non-sharing needs to be justified. We recognise that reproducibility or replicability is not a goal in qualitative research and that, similar to industrial studies, qualitative studies often face challenges in sharing research data".

And also *"submissions must supply all information that is needed to replicate the results, and therefore are expected to include or point to a replication package with the necessary software, data, and instructions. Reviewers may consult these packages to resolve open issues. There can be good reasons for the absence of a replication package, such as confidential code and/or data, the research being mostly qualitative, or the paper being fully self-contained. If a paper does not come with a replication package, authors should comment on its absence in the submission data".*

This strong commitment toward the sharing of software and data (the lack of this information seriously undermines the acceptance of a paper in those venues), even if complex to be accomplished, plays an important role in the improvement and simplification of the research activities, in particular:

- Results replicability: the presented results can be checked by the community, so that they are more reliable.

[3] `https://www.unesco.org/en/open-science`
[4] `http://www.icse-conferences.org/`
[5] `https://www.esec-fse.org/upcoming_events`
[6] `https://icst2022.vrain.upv.es/series/icst`

- Comparability: when a new technique is presented, it can be compared in a simpler way with the previous ones, since it can be tested on the same dataset/testbed the previous ones were tested on.
- Testbed availability: years after years, repository with applications, libraries, source code are created and maintained, so that they can be directly exploited to verify and validate new tools and techniques.
- Dissemination inside the community: the availability of the code of a tool, possibly with test suites and data for testing, makes those research results more known, cited and reused inside the community

On the contrary, the aforementioned points are painful for our community, even if we are moving toward them too (e.g., in the AAMAS 2023 call for paper, a reference can be found to this aspect: *"We highly encourage authors to make their source code (if any) publicly available after their papers are accepted. The link should be in their paper and will also be publicised on the AAMAS website"*). Nonetheless, it is really difficult to find the code (open source or compiled) of proposed frameworks/tools and of "real MASs" to be used to validate new runtime techniques, or to be used as testbed for new testing approaches, or to be analysed in their structure for reverse engineering activities. For example, a simple research on Gitlab through its topics shows that few projects exist under "Multiagent", but also if searching in general on the web, finding MASs that are more than a toy example, or some academic project, well documented, reusable, is quite impossible. So, we often rely on "toy examples" or "MASs developed by the same research group", that is, internal resources, to substantiate our claims. This makes the validation and verification of new tools and techniques weaker: from a SE point of view, this is a clear threat to validity. Also, when searching the state of the art for similar approaches or tools, to perform a comparison, it is often difficult to find an artefact to be used. While we can find works to compare with, rarely a thorough and real comparison can be performed that goes beyond reading the paper. This, sometimes, is determined by the fact that the code used in existing papers is not available to the community. While these aspects do not decrease the intrinsic quality of the research, they limit its visibility and make it less shareable, both inside and outside the community.

To help solving these problems, and to support the sharing of the results of the community, we promoted the "Fantastic MASs and where to find them" initiative, aimed at:

- Promoting the visibility of the results of the research, to facilitate the study of the state of the art, by offering a repository of works, organised in macro areas.
- Promoting the creation of a repository of MASs, to be used as third party testbed and so simplifying the validation and verification of new tools and approaches.
- Promoting the reproducibility of the results, and the sharing of the code of tools, algorithms and so on, so that to simplify the comparison with previous works.

– Supporting the Software Engineering activities for the community, sharing information regarding bugs, test suites and so on associated with already existing MASs and tools.

To create this repository, we decided not to perform a standard Systematic Literature Review (SLR), mainly due to two reasons:

– Some of the information of our interest are not available in papers (e.g., bugs, previous versions, and so on).
– We focus on artefacts to be shared inside the community – not mere theoretical approaches –, with a format that can be reused by others; such information is often missing in papers, or even not published.

So, the idea was to perform a sort of crowdsourcing of information; thus, bottom up (rather than top down, as in a SLR). Our goal is to help the community grow and share existent agent-based technology. This is advantageous both for those contributing to the repository who will have the opportunity to increase the visibility of their work, and for its end users who will find existent works in a simpler way. This initiative may be seen as orthogonal to other top down reviews, like for example [43] that surveyed the literature to evaluate the practical application impact of Multiagent Systems and Technologies (MAST), or for example to the platform AI4europe[7].

The paper is organised as follow: Section 2 describes the process we followed to set up the call of the initiative, Section 3 reports the submissions we collected till the 20th of February 2023, Section 4 presents some considerations and lesson learned, and Section 5 concludes.

2 Selection process

Before reporting the results and discussing them, we linger on the process we followed to set up the repository. Specifically, the main categories we used to classify the contributions, and which kind of questions we asked, and to whom.

The aim of the repository is dual:

1. To offer a collection of MASs to be used as testbed for Verification and Validation (V&V) activities.
2. To offer an overview of available MAS development frameworks, libraries and tools available to the community, to support the study of the state of the art and at the same time offering a simple way to retrieve the code/source code of the identified artefacts, so that to be able to concretely use them for extension, adoption or comparison.

So, the foreseen contributions cover a large variety of types, and some classification was needed both to collect them, and to organise them in the repository. We decided to distinguish three main types of contributions: i) MASs ii) Frameworks and iii) Extensions. Contributions that are labelled as “Framework” are

[7] `https://www.ai4europe.eu/`

completely new agent development frameworks to create MASs (something with the same aim of Jade, Jason and so on), while contributions labelled as "Extensions" are libraries/add-ons for already existing frameworks that improve the latter's capabilities, without turning it into a completely new framework. From this viewpoint, the add-ons that can be found on the Jade webpage[8] are, in our classification, "Extensions", while JACK[34] is a new "Framework".

The collected entries have been further divided into 3 different categories, that are (citing the labels offered in the submission form):

– "Agent-based simulation (you developed a MAS, or a framework/library/ add-on to simulate physical and natural phenomena)": these entries pertain to the Multiagent Based Simulations (MABS) area, so MASs where many agents have been used to **Simulate a specific situation** (like systems frequently developed for example in Netlogo), or frameworks and extensions offering support in this area.
– "Agent-oriented software engineering (you developed a MAS that is the "real" system, for example for implementing decision support systems/ solving industrial problems/implementing smart systems, or a framework/library/ add-on to develop such real MASs)": the MASs in this category are **Systems exploiting the MAS paradigm** to solve a specific task (often developed for example in Jade or JaCaMo), or frameworks and extensions offering some SE related activity in this area.
– "Other": other entries that do not specifically fall in the two previous categories (and in this case, we let the submitter to insert a description).

The category is chosen by the author creating the entry, as all the other information. However, after the first round of call, we realised that some submissions were borderline between the two macro categories or have been labelled with "Other", so we plan to add a further category (as reported below, it could be "V&V tools for MASs") and to explain better how to classify, with respect to the adopted classification, a product to be inserted in the repository.

In the following, we only report the most relevant questions asked to the submitters: for a complete understanding, readers can refer to the website containing the results of our call[9], and the form to submit a new contribution.

For each entry – which can be a MAS, a framework, or an extension – we asked where to find its repository and whether previous repositories exist: the second question is important to better track the history of the entry. Other than that, we also asked for main publications (whether available) where the entry had been firstly introduced.

Then, we asked for additional details on its development: for instance, if any software engineering approach was followed.

Last, but not least, we asked about its being tested. This aspect is important to understand the maturity of the entry. Specifically, we were interested in knowing whether some specific approach had been followed to test the entry, if

[8] `https://jade.tilab.com/download/add-ons/`
[9] `https://mas-unige.github.io/fantastic_mass/`

the test suites were available too, and if a bug/issue tracker/list was available too (and where to find them).

Moreover, only for MASs and extensions, we asked additional questions about its corresponding framework, such as where the extended framework can be found, which version has been considered, and so on.

The second aspect of interest in the selection process is to whom we asked the questions. To be as fair as possible, and at the same time, as general as possible, we submitted a call on the most influential mailing lists in the agent community (e.g., `agents@cs.umbc.edu`, and so on).

The outcome of our call for frameworks, MASs, and extensions is a publicly available repository: `https://mas-unige.github.io/fantastic_mass/`. The repository contains relevant information such as: links to the resources, links to scientific articles, and general information about the development of the resource.

3 Results

In this section, we report the results of our call. In detail, we received 33 submissions, classified as follows:

- 4 implementations (MASs),
- 21 frameworks,
- 8 existing framework extensions.

3.1 MASs

Here, we report the MASs that have been gathered in our call. In total, 4 MASs have been reported; the first two are in the AOSE area, while the others are in the MABS area.

AdaptSchedule [27] is a MAS designed to assist adolescents, particularly adolescents with disabilities, transition towards independent management of their own schedules. It allows them to set up a daily schedule with all activities that must be performed and any constraints between those activities.

MAPS-HOLO [19] proposes a Holonic Multiagent System (HMAS) to assign and manage parking spaces in a smart parking system called Holonic Multiagent Parking System (MAPS-HOLO) developed through the JaCaMo Framework described in Section 3.2. Besides assigning parking space, the system will be able to handle run-time agents failures in different levels: driver agent, sector agent, and manager agent failure.

Deep Q-Learning agents for traffic signal control [57] presents a Reinforcement Learning approach to traffic lights control, coupled with a microscopic agent-based simulator (Simulation of Urban MObility - SUMO) providing a synthetic but realistic environment in which the exploration of the outcome of potential regulation actions can be carried out.

The Affective Agents [31,6] project aims at modeling interactions between people considering their *affective state* (representing their attitude towards other people wearing or not a mask, if they are indoor or outdoor, considering their age and gender and their mood expressed as a mix of fear of the contagious of covid19 and "internal and external social distance"). The combination of these parameters makes the subject to adopt a different "hall space" (that is, the distance from another person that a subject consider safe for him). The data guiding these simulations implemented in Netlogo (3 different models are currently offered) have been collected in the field thanks to two experiments with human subjects.

3.2 Frameworks

We report all the frameworks that have been collected in our call. We split them into four categories: the classification we use in the sequel is not the same we used to guide the selection process. Specifically, in the selection process, we asked the community to classify each entry w.r.t. three possible keywords: "Agent-Oriented Software Engineering", "Agent-Based Simulation", and "Other". However, when reporting the obtained results, another keyword emerged, that is "Verification of MAS". Thus, in the following, we reported the collected frameworks also in terms of this additional label.

Amongst the gathered frameworks, 14 works are in the agent-based software engineering area, 3 are in the verification area, 2 in the agent-based modeling and simulation area, and 2 in another area. Furthermore, considering the supported agent architectures [17], we may find 6 contributions adopting the BDI one, while the others do not follow any specific agent architecture.

Both well-established and newly born frameworks have been reported. Some of them are based upon implemented Domain Specific Languages, DSL.

Agent-oriented software engineering. AgentScript Cross-Compiler (ASC2) [47] is a MAS framework mainly for agents created with the AgentScript agent programming language[10]. The language of ASC2 is based on Jason [15]. The novelty of this framework is in relying on the Actor model, instantiating each intentional agent as an autonomous micro-system run by actors. ASC2 works as a cross-compiler that translates the high level language of AgentScript into lower level executable languages, such as Scala. Possible interactions between the agent-based programming framework ASC2 and various testing approaches (unit/agent testing, integration/system testing, continuous integration) have been extensively evaluated in [46].

ASTRA [21,26] stands for AgentSpeak(TR) Agents and is an Agent-Oriented Programming Language combining AgentSpeak(L) [50] and Teleo Reactive programming [44]. ASTRA is designed to be easy to learn and familiar to developers who are experienced in using mainstream Object-Oriented Programming Languages.

[10] https://github.com/mostafamohajeri/scriptcc-translator

DALI [22] is a meta interpreter built on top of Sicstus Prolog[11]. DALI is an Active Logic Programming language, designed for executable specification of logical agents, without committing to any specific agent architecture. DALI allows the programmer to define one or more agents, interacting among themselves, with an external environment, or with a user.

JaCaMo [14,13] is composed of three technologies, Jason, CArtAgO [51], and Moise [35], each representing a different abstraction level. Jason is used for programming the agent level, CArtAgO is responsible for the environment level, and Moise for the organisation level. JaCaMo integrates these three technologies by defining a semantic link among concepts in different levels of abstraction (agent, environment, and organisation). The end result is the JaCaMo MAS development platform. It provides high-level first-class support for developing agents, environments, and organisations, allowing the development of more complex MASs.

Jade [9,8] is an open source platform for the development of agent based applications. Besides the agent abstraction, it also provides: task execution and composition model, peer-to-peer agent communication based on asynchronous message passing, and a yellow page service that supports the publish and subscribe discovery mechanism. JADE-based systems can be distributed across machines with different operational systems, and has been used by many languages (e.g., Jason and JaCaMo) as a distribution infrastructure.

Jadescript [11,10] is a recent AOP language designed to develop Jade agents. It provides a set of agent-oriented linguistic constructs and related abstractions, namely agents, (agent) behaviours, and (communication) ontologies. Agents written in Jadescript are executed in JADE platforms, and they interact via asynchronous messaging. Therefore, Jadescript adopts an event-driven programming style.

Python Agent DEvelopment (PADE) framework [41] is an open source platform implemented in Python language and conceived for the implementation of MASs on power systems. PADE is compliant with specifications of the Foundation for Intelligent Physical Agents (FIPA) and eases the development of solutions to power systems based on MAS.

Another framework similar to PADE is Python Intelligent Agent Framework (PIAF)[12], which has not been already published.

BSPL [52,55] stands for the Blindingly Simple Protocol Language and is an information-based protocol language. In BSPL, it is possible to describe the communication protocols as well as the corresponding agents' enactment of the latter.

Deserv [53,56] is a protocol-based programming model for decentralized applications that is suited to the cloud. Specifically, Deserv demonstrates how to leverage function-as-a-service (FaaS), a popular serverless programming model, to implement agents.

[11] `https://sicstus.sics.se/`

[12] `https://gitlab.com/ornythorinque/piaf`

Hercule [54] an approach for declaratively specifying blockchain applications in a manner that reflects business contracts. Hercule represents a contract via regulatory norms that capture the involved parties' expectations of one another. It computes the states of norms (hence, of contracts) from events in the blockchain.

JS-son [36] is a lean JavaScript library prototype for implementing reasoning-loop agents. The library focuses on core agent programming concepts and refrains from imposing further restrictions on the programming approach.

SMASTA+ [7] is a Scala implementation of the Extended Multi-agents Situated Task Allocation.

StreamB [29] is a Domain Specific Language (DSL) for processing data streams in abstract environments. With StreamB it is possible to guide the translation from low-level information (e.g. sensors) to high-level concepts (e.g. beliefs). Thanks to its declarative nature, StreamB is much more intuitive and helps the user to create abstract environments, by reducing the amount of actual code to be produced; since the mapping process is automatically synthesised by StreamB. In more detail, StreamB is built upon the notion of Stream Processing, and allows the user for a flexible yet straightforward way to map low-level environment data, to high-level agent beliefs.

Verification of MAS. EVE (Equilibrium Verification Environment) [33,32] is a formal verification tool for the automated analysis of temporal equilibrium properties of concurrent and multiagent systems represented as multi-player games. Systems are modeled using the Simple Reactive Module Language (SRML) as a collection of independent system components (players/agents in a game), which are assumed to have goals expressed using Linear Temporal Logic (LTL) formulae. In particular, EVE checks for the existence of Nash equilibria in such systems and can be used to do rational synthesis and verification automatically.

The MCAPL [23,24] (Model-checking Agent Programming Languages) framework is a suite of tools for building interpreters for agent programming languages and verifying the correctness of programs running in these interpreters using the model checking technique. It consists of the Agent Infrastructure Layer (AIL) toolkit for building interpreters for rational agent programming languages (BDI languages) and the Agent JavaPathFinder (AJPF) model checker [25].

STV [38,37] is a collection of algorithms for verifying Alternating-time Temporal Logic (ATL) properties on models with perfect (resp. imperfect) information, and exploiting imperfect recall strategies.

Agent-Based Simulation. CellNet Network [45] is an open-source Java-based software developed as a research resource to study MAS, evolutionary game theory and cellular automata simulations. CellNet works in two modes: (i) using a graphical user interface (GUI) for doing micro-simulations or (ii) using a batch mode for doing macro-simulations.

Swarm-Like Protocol in Python (SLAPP) [12,40] comes from Swarm [42]. SLAPP is only one of the possible flavors of Swarm; it is a simplified flavor, because it is written in Python.

Others. MatchU [30] is a web-based platform that offers an interactive framework to find how to form mutually-beneficial relationships, decide how to distribute resources, or resolve conflicts through a suite of matching algorithms rooted in economics and artificial intelligence.

MAPF [13] is a collection of techniques and tools to perform Multi-Agent Path Finding (MAPF). Such repository has been submitted to our call, but no specific framework has been pointed out. Thus, the analysis we perform on the other tools cannot be performed on this repository as well. Nonetheless, even though in the following we do not analyse MAPF with the rest of the tools, we recognise its importance and legitimacy in being listed with the other MAS frameworks in our repository.

3.3 Extensions

Here, we report the frameworks' extensions that have been gathered in our call. In total, 8 different extensions have been reported. Of these, 4 are extensions of JaCaMo, 1 is an extension of Cartago, 1 is an extension for Jade, 1 is an extension of SUMO, and 1 is an extension of MCAPL. Since both JaCaMo and MCAPL are based on the BDI architecture, 4 out of 5 extensions can be classified as BDI projects.

2COMM [4] is an extension of the JaCaMo framework for defining social relationships, represented as social commitments, among parties, conceived as autonomous agents.

JaCaMo+Accountability [5] proposes an extension of JaCaMo with the notion of accountability, grounded on responsibility, that supports the development of robust distributed systems.

JaCaMo+Exceptions [3] is an extension of Moise, the organizational model and infrastructure adopted in JaCaMo, that explicitly encompasses the notion of exception as a first-class element in the design of a multiagent organization.

Multi-Agent MicroServices (MAMS) [20] is an architectural style for integrating MASs into Microservices architectures. It extends Cartago and achieves this by modeling agents as entities that have hypermedia bodies that are exposed as REST APIs. This provides a standard REST API that plain-old microservices can exploit.

ROS-A [18] an interface for integrating BDI-based agents into robotic systems developed using ROS [49]. The authors use the Gwendolen language to program the BDI agents and to make use of the AJPF model checker in order to verify properties related to the decision-making in the agent programs.

[13] `http://mapf.info`

SUMO-RL [1] provides a simple interface to instantiate Reinforcement Learning environments with SUMO[14] for Traffic Signal Control. SUMO-RL provides a simple interface to instantiate Reinforcement Learning environments with SUMO for Traffic Signal Control. Goals of this repository: (i) provide a simple interface to work with Reinforcement Learning for Traffic Signal Control using SUMO (ii); support Multiagent RL; (iii) compatibility with gym.Env and popular RL libraries such as stable-baselines3 and RLlib; (iv) easy customisation: state and reward definitions are easily modifiable.

LEARN [16] is an extension of the standard JADE platform, enhancing it with peer-to-peer (p2p) capabilities. The general idea of LEARN is offer a set of specific agents (JADE agents to be created on a standard JADE platform) able to create a logic p2p layer over a set of JADE platforms, so that to make them able to dynamically discover other platforms without knowing their IP (so in a real p2p fashion) and then invoking services over the p2p layer.

RV4JaCa [28] is an extension of JaCaMo which allows performing runtime verification of agents' messages. It represents the agent-based istantiation of RML (Runtime Monitoring Language) [2], a simple but powerful Domain Specific Language (DSL) for runtime verification.

4 Discussion and lessons learned

In this section, we analyse and discuss the results obtained in our call, focusing on the aspects strictly related to the software engineering area: since the collected MASs are few (which is already a reason for reflection), we limit this analysis to the framework and extension entries. With this analysis we pay attention not on the functionalities of the proposed frameworks and extensions (we will refer to them with the term "tool" in the remaining of the section), but on the aspects concerning their re-usability as subjects for future research.

Table 1 reports the results of our analysis for 27 out of 29 collected tools. In detail, each tool is analysed w.r.t. six different features:

- *Previous Versions*: This feature concerns the presence of previous versions and/or commits on the tool's repository. This aspect is important to better understand and study the tool's evolution, like for example when searching for architectural smells, as done in [48].
- *Documentation*: This feature is about the availability of documentation supporting the tool. This is of paramount importance to increase the tool's usability. Furthermore, a good documentation can be exploited for example for automatic oracles extraction, or reverse engineering tasks.
- *Issue and Bug tracker*: This feature concerns the presence in the tool's repository of well-documented issues and bugs, possibly with information regarding if, and how, they were fixed. This kind of information is used in SE to validate, for example, tools for automatic bugs identification.

[14] `https://github.com/eclipse/sumo`

Subject	Ref.	Versioning	Document.	Issue-tracker	Test	Linked	Source
ASC2	[47]	✓	✓	✗	(✓)	✓	✓
ASTRA	[21,26]	✓	✓	✓	✓	✓	✓
BSPL	[52,55]	✓	✓	✗	✓	✓	✓
CellNet	[45]	✗	✓	✗	✗	✗	✓
DALI	[22]	✓	✓	✓	(✓)	✗	✓
Deserv	[53,56]	✓	✓	✗	(✓)	✓	✓
EVE	[33,32]	✓	✓	✗	✗	✓	✓
Hercule	[54]	✓	✓	✗	✓	✓	✓
JaCaMo	[14,13]	✓	✓	✗	✗	✓	✓
Jade	[9,8]	✓	✓	(✓)	✓	✓	✓
Jadescript	[11,10]	✓	✓	✓	✓	✗	✓
JS-son	[36]	✓	✓	(✓)	✓	✓	✓
MCAPL	[23,24]	✓	✓	✗	✗	✓	✓
PADE	[41]	✓	✓	✗	✗	✓	✓
Piaf		✓	✓	✓	✓	✗	✓
SLAPP	[12,40]	✓	✓	✓	✗	✓	✓
SMASTA+	[7]	✓	✓	✗	✗	✓	✓
StreamB	[29]	✓	✓	✗	✗	✓	✓
STV	[38,37]	✓	✓	✗	✗	✗	✓
2COMM	[4]	✓	✗	✗	✗	✓	✓
JaCaMo+Acc.	[5]	✓	✓	✗	✗	✗	✓
JaCaMo+Exc.	[3]	✓	✓	✗	✗	✓	✓
LEARN	[16]	✗	✓	✗	✗	✗	✓
MAMS	[20]	✓	✓	✗	✗	✗	✓
ROS-A	[18]	✓	✓	✗	✗	✓	✓
RV4JaCa	[28]	✓	✓	✗	✗	✓	✓
SUMO-RL	[1]	✓	✓	✓	✗	✗	✓
Total:	27	25✓ 0(✓) 2✗	26✓ 0(✓) 1✗	6✓ 2(✓) 19✗	7✓ 3(✓) 17✗	18✓ 0(✓) 9✗	27✓ 0(✓) 0✗

Table 1. Summary of the submissions (Frameworks and Extensions), excluding MAPF and MatchU, that cannot be compared based on these features.

– *Test*: This feature is about the availability of tests, or at least information regarding the testing approach used to validate the framework. Test suites that can be downloaded along with the tool can be used, for example, for regression testing, or to validate tools for automatic test generation.
– *Original Link*: In this column we report if a link to the code (source or compiled, stored on a public repository on online so that it is downloadable) was already present in the principal papers (as indicated by the author who created the entry in our repository) or not
– *Source Code*: All the entries collected on our repository link to a downloadable version of the tool, but the availability of the source code is optional. The source code of the tool, with an open licence to reuse it, makes it simply to be reused by other researchers both for extending, both for other activities, like for example static analysis

For each feature, we report ✓(resp., ✗) when a tool offers (resp., does not offer) such a feature. Moreover, in case the feature is partially supported, the symbol (✓) is used. For instance, considering the documentation feature, the symbol (✓) is reported if the tool offers some documentation, but, the latter is minimal. The last row of the table reports the count of ✓, (✓) and ✗, so that the reader can simply have an overview of the results.

Note that, in Table 1, MatchU [30] is not reported since it is a web service: none of the previously listed features is offered. For a different reason, as mentioned previously, also MAPF is not reported in Table 1; indeed, MAPF is not a framework, but a collection of frameworks. Thus, a comparison w.r.t. the features of interest would have been unnatural.

Considering the column *Source*, clearly there is a general positive propensity to share the source code, which makes these tools usable for further extensions or analysis. Anyway, as shown in column *Linked*, the source code was not always originally shared with the paper presenting the work (even in important venues like AAMAS or international journals), but was made public in a second moment[15]. This shows that the sharing of the tool itself with the paper is not yet so widely adopted by the community. Nevertheless, the shared software is usually well documented.

An interesting result from the analysis is that quite all the shared tools (all but two) present a form of versioning, that allow users to access previous versions and commits: this may be of interest for those working with software evolution.

A clear problematic aspect is the one related to the management of bugs and issues, and the usage and sharing of test suites: only seven tools reported some information related to the availability of a test suite or something similar, and again only seven tools (interestingly, not all the same reporting information regarding test suites) refer to a bug/issue tracker. This is anyway not surprising: the management of bugs and issues is a complex task, usually requesting the support of specific tools, some dedicated resource (tester) or at least time. If the project is not yet very large, does not have a large community interested in it and supporting it, or is primarily academic, the burden requested for managing such aspects may be not manageable. Unfortunately, not having a list of known bugs/issues, associated with a specific version of a tool, makes the SE operations like automatic test generation or self healing hard to be performed (in the sense that, if we miss the bugs history it is impossible to verify if the new technique is able to identify known bugs, which would be the ground truth), or not really effective.

Furthermore, the reality gap between frameworks, and their possible real-world uses, can also be observed in the answers we obtained. Specifically, for each framework, we asked about existing real-world uses. By doing so, we observed that the majority of frameworks do not have (to the best of their creators' knowledge) any real-world application; indeed, the majority of the collected frameworks are mainly used in academia. Nonetheless, some frameworks reported existing real-world applications, such as ASC2 [47], that has been used by the TNO Netherlands Asser Institute[16], or DALI [22], that has been used by a company[17].

[15] To answer the call for creating the "Fantastic MASs" repository, or simply in a different moment from the publication.

[16] `https://www.asser.nl`

[17] SPEE Srl (`https://www.spee.it`).

Last but not least, we only received 4 submissions (on a total of 33) regarding real MASs, which were unexpectedly few with respect to our expectations: surely it could be only a case, and maybe considering a longer time to collect submissions this proportion could change, or we could have not be effective in reaching all the community, but the initial feeling is that the community prefers sharing frameworks and extensions instead of MASs. This could be due to the fact that often a MAS itself (or a system) is seen not as a research result, but more as a way to exemplify some new tool/approach etc. Even if this thought may have some true aspects (considering the academic area and our community, where an important focus is given to new languages, models, architectures and so on, so, some more theoretic results), this mindset leads to not giving sometime the correct importance to the produced software itself (and this is related to the often missing link to the code too, discussed previously). Another motivation could be related to the "foreseen time to live" of a MAS: if a framework/extension could be ideally born after many months/years of research and would remain the subject of future research, so involving many people in a long lifespan, the development of a MAS may be a work that is self contained (it is developed when it is needed, to solve a specific problem, and then it is done), and its developers may also be no longer available in future time. For this reason, sharing it (and consequently maintaining it a little) may be itself a demanding task, not perceived providing a valuable "return of investment". Anyway, we hope that this initiative supports a change in the mindset, since as said in the introduction, the unavailability of real MASs prevent from performing unbiased activities of V&V.

5 Conclusions and Future Work

In this paper, we reported the results obtained by asking the agent community about existing frameworks for developing MASs, their extensions (libraries, add ons and similar), and MASs. We presented our selection process, and we briefly summarised each so collected work. The outcome of this work is a publicly available repository, where all the collected works can be easily accessed. Other than serving as a common place for retrieving agent-based frameworks, their extensions, and MAS models, the repository serves as a milestone to keep track of relevant engineering information. This last aspect is going to be important to maintain a historical memory on agent technologies, and their engineering.

Future work includes maintaining and improving the "Fantastic MASs" repository: for example, as emerged by the analysis of the submissions, it could be better to add a new category regarding V&V. Also, we plan to add an interface for querying and ordering the results, to facilitate the usage of the repository.

Moreover, to keep track of new developments, and existing ones that did not participate in our call, we are going to let researchers propose new entries in the repository, keeping the form to submit new entries always available on the website of the repository. We also hope that sharing these first results with the community will help in getting new submissions. This will allow the repository to remain updated, and to properly resemble the current agent-based technology ecosystem.

As a final activity, the already mentioned European AI-on-demand (AIOD) platform, `https://www.ai4europe.eu/`, seeks to bring together the AI community and act as facilitator of knowledge transfer from research to multiple business domains. Making the results of the “Fantastic MAS” initiative available on that platform, or just linked from there, would make them more visible and more useful, also outside the Engineering MAS community.

Acknowledgements

We thank Olivier Boissier, Rafael Heitor Bordini, Jomi Fred Hübner, and Giuseppe Vizzari for the precious feedback they provided while designing the “Fantastic MAS” submission form, and all the contributors.

References

1. Alegre, L.N.: SUMO-RL. `https://github.com/LucasAlegre/sumo-rl` (2019)
2. Ancona, D., Franceschini, L., Ferrando, A., Mascardi, V.: RML: theory and practice of a domain specific language for runtime verification. Sci. Comput. Program. **205**, 102610 (2021). https://doi.org/10.1016/j.scico.2021.102610, `https://doi.org/10.1016/j.scico.2021.102610`
3. Baldoni, M., Baroglio, C., Boissier, O., Micalizio, R., Tedeschi, S.: Distributing responsibilities for exception handling in jacamo. In: Dignum, F., Lomuscio, A., Endriss, U., Nowé, A. (eds.) AAMAS ’21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021. pp. 1752–1754. ACM (2021). https://doi.org/10.5555/3463952.3464226, `https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p1752.pdf`
4. Baldoni, M., Baroglio, C., Capuzzimati, F., Micalizio, R.: Commitment-based agent interaction in jacamo+. Fundam. Informaticae **159**(1-2), 1–33 (2018). https://doi.org/10.3233/FI-2018-1656, `https://doi.org/10.3233/FI-2018-1656`
5. Baldoni, M., Baroglio, C., Micalizio, R., Tedeschi, S.: Robustness based on accountability in multiagent organizations. In: Dignum, F., Lomuscio, A., Endriss, U., Nowé, A. (eds.) AAMAS ’21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021. pp. 142–150. ACM (2021). https://doi.org/10.5555/3463952.3463975, `https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p142.pdf`
6. Bandini, S., Briola, D., Dennunzio, A., Gasparini, F., Giltri, M., Vizzari, G.: Integrating the implications of distance-based affective states in cellular automata pedestrian simulation. In: Chopard, B., Bandini, S., Dennunzio, A., Haddad, M.A. (eds.) Cellular Automata - 15th International Conference on Cellular Automata for Research and Industry, ACRI 2022, Geneva, Switzerland, September 12-15, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13402, pp. 259–270. Springer (2022). https://doi.org/10.1007/978-3-031-14926-9_23, `https://doi.org/10.1007/978-3-031-14926-9_23`
7. Beauprez, E., Caron, A., Morge, M., Routier, J.: A multi-agent negotiation strategy for reducing the flowtime. In: Rocha, A.P., Steels, L., van den Herik, H.J.

(eds.) Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021, Volume 1, Online Streaming, February 4-6, 2021. pp. 58–68. SCITEPRESS (2021). https://doi.org/10.5220/0010226000580068, https://doi.org/10.5220/0010226000580068

8. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE: A software framework for developing multi-agent applications. lessons learned. Inf. Softw. Technol. **50**(1-2), 10–21 (2008). https://doi.org/10.1016/j.infsof.2007.10.008, https://doi.org/10.1016/j.infsof.2007.10.008
9. Bellifemine, F., Poggi, A., Rimassa, G.: Developing multi-agent systems with a fipa-compliant agent framework. Softw. Pract. Exp. **31**(2), 103–128 (2001). https://doi.org/10.1002/1097-024X(200102)31:2¡103::AID-SPE358¿3.0.CO;2-O, https://doi.org/10.1002/1097-024X(200102)31:2<103::AID-SPE358>3.0.CO;2-O
10. Bergenti, F., Caire, G., Monica, S., Poggi, A.: The first twenty years of agent-based software development with JADE. Auton. Agents Multi Agent Syst. **34**(2), 36 (2020). https://doi.org/10.1007/s10458-020-09460-z, https://doi.org/10.1007/s10458-020-09460-z
11. Bergenti, F., Monica, S., Petrosino, G.: A scripting language for practical agent-oriented programming. In: Koster, J.D., Bergenti, F., Franco, J. (eds.) Proceedings of the 8th ACM SIGPLAN International Workshop on Programming Based on Actors, Agents, and Decentralized Control, AGERE!@SPLASH 2018, Boston, MA, USA, November 5, 2018. pp. 62–71. ACM (2018). https://doi.org/10.1145/3281366.3281367, https://doi.org/10.1145/3281366.3281367
12. Boero, R., Morini, M., Sonnessa, M., Terna, P., Terna, P.: Introducing the swarm-like agent protocol in python (slapp). Agent-based Models of the Economy: From Theories to Applications pp. 31–54 (2015)
13. Boissier, O., Bordini, R., Hubner, J., Ricci, A.: Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo. Intelligent Robotics and Autonomous Agents series, MIT Press (2020), https://books.google.com.br/books?id=GM_tDwAAQBAJ
14. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with jacamo. Sci. Comput. Program. **78**(6), 747–761 (2013). https://doi.org/10.1016/j.scico.2011.10.004, https://doi.org/10.1016/j.scico.2011.10.004
15. Bordini, R.H., Wooldridge, M., Hübner, J.F.: Programming Multi-Agent Systems in AgentSpeak using Jason. John Wiley & Sons (2007)
16. Briola, D., Micucci, D., Mariani, L.: A platform for P2P agent-based collaborative applications. Softw. Pract. Exp. **49**(3), 549–558 (2019). https://doi.org/10.1002/spe.2657, https://doi.org/10.1002/spe.2657
17. Cardoso, R.C., Ferrando, A.: A review of agent-based programming for multi-agent systems. Comput. **10**(2), 16 (2021). https://doi.org/10.3390/computers10020016, https://doi.org/10.3390/computers10020016
18. Cardoso, R.C., Ferrando, A., Dennis, L.A., Fisher, M.: An interface for programming verifiable autonomous agents in ROS. In: Bassiliades, N., Chalkiadakis, G., de Jonge, D. (eds.) Multi-Agent Systems and Agreement Technologies - 17th European Conference, EUMAS 2020, and 7th International Conference, AT 2020, Thessaloniki, Greece, September 14-15, 2020, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12520, pp. 191–205. Springer (2020). https://doi.org/10.1007/978-3-030-66412-1_13, https://doi.org/10.1007/978-3-030-66412-1_13

19. de Castro, L., Borges, A.P., Alves, G.V., Grossa, C.P.: Developing a smart parking solution based on a holonic multiagent system using jacamo framework. In: Proceedings of the 12th workshop-school on agents, environments, and applications, Fortaleza-CE, Brazil (2018)
20. Collier, R.W., O'Neill, E., Lillis, D., O'Hare, G.M.P.: MAMS: multi-agent microservices*. In: Amer-Yahia, S., Mahdian, M., Goel, A., Houben, G., Lerman, K., McAuley, J.J., Baeza-Yates, R., Zia, L. (eds.) Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019. pp. 655–662. ACM (2019). https://doi.org/10.1145/3308560.3316509, `https://doi.org/10.1145/3308560.3316509`
21. Collier, R.W., Russell, S.E., Lillis, D.: Reflecting on agent programming with agentspeak(l). In: Chen, Q., Torroni, P., Villata, S., Hsu, J.Y., Omicini, A. (eds.) PRIMA 2015: Principles and Practice of Multi-Agent Systems - 18th International Conference, Bertinoro, Italy, October 26-30, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9387, pp. 351–366. Springer (2015). https://doi.org/10.1007/978-3-319-25524-8_22, `https://doi.org/10.1007/978-3-319-25524-8_22`
22. Costantini, S., Tocchio, A., Verticchio, A.: Communication and trust in the DALI logic programming agent-oriented language. Intelligenza Artificiale **2**(1), 39–46 (2005)
23. Dennis, L.A.: The MCAPL framework including the agent infrastructure layer an agent java pathfinder. J. Open Source Softw. **3**(24), 617 (2018). https://doi.org/10.21105/joss.00617, `https://doi.org/10.21105/joss.00617`
24. Dennis, L.A., Fisher, M., Lincoln, N., Lisitsa, A., Veres, S.M.: Practical verification of decision-making in agent-based autonomous systems. Autom. Softw. Eng. **23**(3), 305–359 (2016). https://doi.org/10.1007/s10515-014-0168-9, `https://doi.org/10.1007/s10515-014-0168-9`
25. Dennis, L.A., Fisher, M., Webster, M.P., Bordini, R.H.: Model checking agent programming languages. Autom. Softw. Eng. **19**(1), 5–63 (2012). https://doi.org/10.1007/s10515-011-0088-x, `https://doi.org/10.1007/s10515-011-0088-x`
26. Dhaon, A., Collier, R.W.: Multiple inheritance in agentspeak(l)-style programming languages. In: Boix, E.G., Haller, P., Ricci, A., Varela, C.A. (eds.) Proceedings of the 4th International Workshop on Programming based on Actors Agents & Decentralized Control, AGERE! 2014, Portland, OR, USA, October 20, 2014. pp. 109–120. ACM (2014). https://doi.org/10.1145/2687357.2687362, `https://doi.org/10.1145/2687357.2687362`
27. Durfee, E.H., Garrett, L.H., Johnson, A.: Promoting independence with a schedule management assistant that anticipates disruptions. J. Heal. Informatics Res. **4**(1), 19–49 (2020). https://doi.org/10.1007/s41666-019-00060-5, `https://doi.org/10.1007/s41666-019-00060-5`
28. Engelmann, D.C., Ferrando, A., Panisson, A.R., Ancona, D., Bordini, R.H., Mascardi, V.: Rv4jaca - runtime verification for multi-agent systems. In: Cardoso, R.C., Ferrando, A., Papacchini, F., Askarpour, M., Dennis, L.A. (eds.) Proceedings of the Second Workshop on Agents and Robots for reliable Engineered Autonomy, AREA@IJCAI-ECAI 2022, Vienna, Austria, 24th July 2022. EPTCS, vol. 362, pp. 23–36 (2022). https://doi.org/10.4204/EPTCS.362.5, `https://doi.org/10.4204/EPTCS.362.5`
29. Ferrando, A., Papacchini, F.: Streamb: A declarative language for automatically processing data streams in abstract environments for agent platforms. In: Alechina,

N., Baldoni, M., Logan, B. (eds.) Engineering Multi-Agent Systems - 9th International Workshop, EMAS 2021, Virtual Event, May 3-4, 2021, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13190, pp. 114–136. Springer (2021). https://doi.org/10.1007/978-3-030-97457-2_7, https://doi.org/10.1007/978-3-030-97457-2_7

30. Ferris, J., Hosseini, H.: Matchu: An interactive matching platform. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. pp. 13606–13607. AAAI Press (2020), https://ojs.aaai.org/index.php/AAAI/article/view/7090
31. Giltri, M., Bandini, S., Gasparini, F., Briola, D.: Furthering an agent-based modeling approach introducing affective states based on real data. In: Bazzan, A.L.C., Dusparic, I., Lujak, M., Vizzari, G. (eds.) Twelfth International Workshop on Agents in Traffic and Transportation co-located with the the 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence (IJCAI-ECAI 2022), Vienna, Austria, July 25, 2022. CEUR Workshop Proceedings, vol. 3173, pp. 124–136. CEUR-WS.org (2022), http://ceur-ws.org/Vol-3173/9.pdf
32. Gutierrez, J., Najib, M., Perelli, G., Wooldridge, M.J.: EVE: A tool for temporal equilibrium analysis. In: Lahiri, S.K., Wang, C. (eds.) Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11138, pp. 551–557. Springer (2018). https://doi.org/10.1007/978-3-030-01090-4_35, https://doi.org/10.1007/978-3-030-01090-4_35
33. Gutierrez, J., Najib, M., Perelli, G., Wooldridge, M.J.: Automated temporal equilibrium analysis: Verification and synthesis of multi-player games. Artif. Intell. **287**, 103353 (2020). https://doi.org/10.1016/j.artint.2020.103353, https://doi.org/10.1016/j.artint.2020.103353
34. Howden, N., Rönnquist, R., Hodgson, A., Lucas, A.: Jack intelligent agents – summary of an agent infrastructure. In: Proceedings of the 5th ACM International Conference on Autonomous Agents (2001)
35. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multiagent systems using the moise$^+$ model: programming issues at the system and agent levels. Int. J. Agent Oriented Softw. Eng. **1**(3/4), 370–395 (2007). https://doi.org/10.1504/IJAOSE.2007.016266, https://doi.org/10.1504/IJAOSE.2007.016266
36. Kampik, T., Nieves, J.C.: Js-son - A lean, extensible javascript agent programming library. In: Dennis, L.A., Bordini, R.H., Lespérance, Y. (eds.) Engineering Multi-Agent Systems - 7th International Workshop, EMAS 2019, Montreal, QC, Canada, May 13-14, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12058, pp. 215–234. Springer (2019). https://doi.org/10.1007/978-3-030-51417-4_11, https://doi.org/10.1007/978-3-030-51417-4_11
37. Kurpiewski, D., Mikulski, L., Jamroga, W.: STV+AGR: towards verification of strategic ability using assume-guarantee reasoning. In: Aydogan, R., Criado, N., Lang, J., Sánchez-Anguix, V., Serramia, M. (eds.) PRIMA 2022: Principles and Practice of Multi-Agent Systems - 24th International Conference, Valencia, Spain, November 16-18, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13753, pp. 691–696. Springer (2022). https://doi.org/10.1007/978-3-031-21203-1_47, https://doi.org/10.1007/978-3-031-21203-1_47

38. Kurpiewski, D., Pazderski, W., Jamroga, W., Kim, Y.: Stv+reductions: Towards practical verification of strategic ability using model reductions. In: Dignum, F., Lomuscio, A., Endriss, U., Nowé, A. (eds.) AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021. pp. 1770–1772. ACM (2021). https://doi.org/10.5555/3463952.3464232, https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p1770.pdf
39. Mascardi, V., Weyns, D., Ricci, A.: Engineering multi-agent systems: State of affairs and the road ahead. ACM SIGSOFT Softw. Eng. Notes **44**(1), 18–28 (2019). https://doi.org/10.1145/3310013.3310035, https://doi.org/10.1145/3310013.3310035
40. Mazzoli, M., Morini, M., Terna, P.: Rethinking macroeconomics with endogenous market structure. Cambridge University Press (2019)
41. Melo, L.S., Sampaio, R.F., Leão, R.P.S., Barroso, G.C., Bezerra, J.R.: Python-based multi-agent platform for application on power grids. International Transactions on Electrical Energy Systems **29**(6), e12012 (2019)
42. Minar, N., Burkhart, R., Langton, C., Askenazi, M., et al.: The swarm simulation system: A toolkit for building multi-agent simulations. Santa Fe Institute Working Paper (1996)
43. Müller, J.P., Fischer, K.: Application Impact of Multi-agent Systems and Technologies: A Survey, pp. 27–53. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
44. Nilsson, N.J.: Teleo-reactive programs for agent control. J. Artif. Intell. Res. **1**, 139–158 (1994). https://doi.org/10.1613/jair.30, https://doi.org/10.1613/jair.30
45. Ombuki, B.M.: Juan c. burguillo: Self-organizing coalitions for managing complexity. Genet. Program. Evolvable Mach. **21**(1-2), 263–264 (2020). https://doi.org/10.1007/s10710-019-09372-2, https://doi.org/10.1007/s10710-019-09372-2
46. Parizi, M.M., Sileno, G., van Engers, T.M.: Seamless integration and testing for MAS engineering. In: Alechina, N., Baldoni, M., Logan, B. (eds.) Engineering Multi-Agent Systems - 9th International Workshop, EMAS 2021, Virtual Event, May 3-4, 2021, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13190, pp. 254–272. Springer (2021). https://doi.org/10.1007/978-3-030-97457-2_15, https://doi.org/10.1007/978-3-030-97457-2_15
47. Parizi, M.M., Sileno, G., van Engers, T.M., Klous, S.: Run, agent, run! architecture and benchmarking of actor-based agents. In: Castegren, E., Koster, J.D., Schmidt, T.C. (eds.) AGERE 2020: Proceedings of the 10th ACM SIGPLAN International Workshop on Programming Based on Actors, Agents, and Decentralized Control, Virtual Event, USA, November 17, 2020. pp. 11–20. ACM (2020). https://doi.org/10.1145/3427760.3428339, https://doi.org/10.1145/3427760.3428339
48. Pigazzini, I., Briola, D., Fontana, F.A.: Architectural technical debt of multi-agent systems development platforms. In: Calegari, R., Ciatto, G., Denti, E., Omicini, A., Sartor, G. (eds.) Proceedings of the 22nd Workshop "From Objects to Agents", Bologna, Italy, September 1-3, 2021. CEUR Workshop Proceedings, vol. 2963, pp. 1–13. CEUR-WS.org (2021), http://ceur-ws.org/Vol-2963/paper13.pdf
49. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.: ROS: an open-source robot operating system. In: Workshop on Open Source Software at the International Conference on Robotics and Automation. IEEE, Japan (2009)

50. Rao, A.S.: Agentspeak(l): BDI agents speak out in a logical computable language. In: de Velde, W.V., Perram, J.W. (eds.) Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, January 22-25, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1038, pp. 42–55. Springer, Berlin, Heidelberg (1996). https://doi.org/10.1007/BFb0031845, https://doi.org/10.1007/BFb0031845
51. Ricci, A., Piunti, M., Viroli, M., Omicini, A.: Environment programming in CArtAgO. In: Multi-Agent Programming: Languages, Tools and Applications, chap. 8, pp. 259–288. Multiagent Systems, Artificial Societies, and Simulated Organizations, Springer (2009). https://doi.org/10.1007/978-0-387-89299-3_8
52. V., S.H.C., Chopra, A.K., Singh, M.P.: Bungie: Improving fault tolerance via extensible application-level protocols. Computer **54**(5), 44–53 (2021). https://doi.org/10.1109/MC.2021.3052147, https://doi.org/10.1109/MC.2021.3052147
53. V., S.H.C., Chopra, A.K., Singh, M.P.: Deserv: Decentralized serverless computing. In: Chang, C.K., Daminai, E., Fan, J., Ghodous, P., Maximilien, M., Wang, Z., Ward, R., Zhang, J. (eds.) 2021 IEEE International Conference on Web Services, ICWS 2021, Chicago, IL, USA, September 5-10, 2021. pp. 51–60. IEEE (2021). https://doi.org/10.1109/ICWS53863.2021.00020, https://doi.org/10.1109/ICWS53863.2021.00020
54. V., S.H.C., Chopra, A.K., Singh, M.P.: Hercule: Representing and reasoning about norms as a foundation for declarative contracts over blockchain. IEEE Internet Comput. **25**(4), 67–75 (2021). https://doi.org/10.1109/MIC.2021.3080982, https://doi.org/10.1109/MIC.2021.3080982
55. V., S.H.C., Chopra, A.K., Singh, M.P.: Mandrake: multiagent systems as a basis for programming fault-tolerant decentralized applications. Auton. Agents Multi Agent Syst. **36**(1), 16 (2022). https://doi.org/10.1007/s10458-021-09540-8, https://doi.org/10.1007/s10458-021-09540-8
56. V., S.H.C., Smirnova, D., Chopra, A.K., Singh, M.P.: Protocols over things: A decentralized programming model for the internet of things. Computer **53**(12), 60–68 (2020). https://doi.org/10.1109/MC.2020.3023887, https://doi.org/10.1109/MC.2020.3023887
57. Vidali, A., Crociani, L., Vizzari, G., Bandini, S.: A deep reinforcement learning approach to adaptive traffic lights management. In: Bergenti, F., Monica, S. (eds.) Proceedings of the 20th Workshop ”From Objects to Agents”, Parma, Italy, June 26th-28th, 2019. CEUR Workshop Proceedings, vol. 2404, pp. 42–50. CEUR-WS.org (2019), http://ceur-ws.org/Vol-2404/paper07.pdf