computers

MDPI

Review

A Review of Agent-Based Programming for Multi-Agent Systems

Rafael C. Cardoso * and Angelo Ferrando *

Department of Computer Science, The University of Manchester, Manchester M13 9PL, UK
* Correspondence: rafael.cardoso@manchester.ac.uk (R.C.C.); angelo.ferrando@manchester.ac.uk (A.F.)

Abstract: Intelligent and autonomous agents is a subarea of symbolic artificial intelligence where these agents decide, either reactively or proactively, upon a course of action by reasoning about the information that is available about the world (including the environment, the agent itself, and other agents). It encompasses a multitude of techniques, such as negotiation protocols, agent simulation, multi-agent argumentation, multi-agent planning, and many others. In this paper, we focus on agent programming and we provide a systematic review of the literature in agent-based programming for multi-agent systems. In particular, we discuss both veteran (still maintained) and novel agent programming languages, their extensions, work on comparing some of these languages, and applications found in the literature that make use of agent programming.

Keywords: agent-based programming; multi-agent systems; agent programming languages

Citation: Cardoso, R.C.; Ferrando, A. A Review of Agent-Based Programming for Multi-Agent Systems. *Computers* **2021**, *10*, 16. https://doi.org/10.3390/computers10020016

Academic Editor: Paolo Bellavista
Received: 15 January 2021
Accepted: 22 January 2021
Published: 27 January 2021

1. Introduction

Multi-Agent Systems (MASs) [1] are a well established branch of Artificial Intelligence (AI). Even though they are relatively young with respect to more archetypal research areas, MASs have a rich history; in 1995 [2] agent technology was recognised as a rapidly developing research area and one of the fastest growing areas of information technology. Such a statement is still true nowadays, where one can find plenty of research articles, tools, and conferences whose aim is to advance the research in the area. Despite this, MASs are not as widely used as they could be. Considering the agent programming aspect of MASs, according to [3], the key reason is that there is little incentive for developers to switch to current Agent Programming Languages (APLs), as the behaviours that can be easily programmed are sufficiently simple to be implementable in mainstream languages with only a small overhead in coding time. This, amongst the presence of too many unorganised options available, does not help agent-based programming languages and tools to be picked from non-expert users.

An intelligent agent [4] can be generalised as a computerised entity that: is able to reason (rational/cognitive), to make its own decisions independently (autonomous), to collaborate with other agents when necessary (social), to perceive the context in which it operates and react to it appropriately (reactive), and finally, to take action in order to achieve its goals (proactive). An agent-based (or agent-oriented) system is a system where the agents are the main entities, treated as first-class abstractions. From a programming perspective, the same reasoning can be followed. In particular, by using a comparison, we can say that agents are to Agent-Oriented Programming (AOP) languages as objects are to Object-Oriented Programming (OOP) languages. In an agent-based programming language, agents are the building blocks, and programs are obtained by programming their behaviours (how an agent reasons), their goals (what an agent aims to achieve) and their interoperation (how agents collaborate to solve a task).

Agents are well-suited to be used in applications involving distributed or concurrent computation or when communication is required between different components. For this reason, agent technology is useful in applications that reason about messages/objects

received over a network. By preserving their processing state and the state of the world around them, agents are also ideally suited to automation applications. Moreover, autonomous agents can operate without user intervention and can be used in applications such as plant/process automation, workflow management, robotics, and others. Another advantage of agent-based programming is that due to the reasoning cycle present in agents, it is also possible to provide explanations about the decisions that an agent has made.

The aim of this paper is to review the latest work in agent-based programming, to help both experts and non-experts users having a better grasp over the current state of the art in agent-based programming technologies, and to identify future directions of research in this area. In particular, we focus on the latest agent programming languages, platforms and frameworks for the development of MASs. Both theoretical and practical papers are taken into account, and we also briefly discuss recent extensions, existing comparisons, and applications of agent-base programming.

With respect to other reviews and surveys on APLs in literature [3,5–10], our review focuses on recent developments and considers works presenting new APLs, as well as works focusing on extending or comparing existing APLs. Moreover, we consider both theoretical and practical aspects; this helps to have a better understanding of the reality gap between theoretical and practical APLs. Finally, with respect to previous reviews, we focus on the entire class of APLs, and not on a specific area of APLs as in [6], where only the engineering aspects are considered, or in [7], where only APLs platforms are considered, or in [8], where only agent-based simulation literature is analysed, or in [3,5], where their focus is in a specific model of agency (Belief-Desire-Intention—BDI).

This paper is structured as follows. A brief history on agent-based programming is given in Section 2. In Section 3, the systematic review process followed in this paper is presented. Section 4 contains the review findings with the papers found in our search of the literature. In Section 5, the review's results are discussed and future directions are suggested. Finally, in Section 6, the conclusions of the paper are reported.

2. History on Agent-Based Programming

In 1993, Agent-Oriented Programming was first introduced [11] as a specialisation of Object-Oriented Programming. Most notably, it discusses the notion of the mental state of an agent, consisting of its information, decisions, and capabilities. This work also describes agent programs in the AGENT-0 interpreter (implemented in the Lisp language) and their communication using speech act theory, the latter is still used to define agent communication in several contemporary agent programming languages. Over the years, many reasoning and cognitive models have been developed for agent-based programming. In this section, we discuss three particular models that have been fundamental in the design of many agent programming languages in the past and that are still being used in new languages these days: Procedural Reasoning System (PRS), BDI, and Situation Calculus.

The Procedural Reasoning System (PRS) [12] (implemented in Lisp) defines a system capable of reasoning about processes, that is, procedural forms of knowledge. An agent in this system is then able to use these procedures to select intentions for achieving particular goals. Unlike in conventional programming languages, these procedures are not invoked a priori, but they are triggered when they are able to contribute towards some goal or to react to some situation. While sharing some similarities to AI planners of the time, its main difference is that it performs partial hierarchical planning in the sense that it interacts with a dynamic environment during the reasoning process, instead of generating a plan for a static environment.

The Belief-Desire-Intention (BDI) model [13,14] consists of a reasoning process that aids the decision-making of selecting an appropriate action towards the achievement of some goal. Its three mental attitudes are: belief—knowledge that the agent believes about its environment, itself, and other agents; desire—the desired states that the agent wants to achieve; and intention—a sequence of steps towards the achievement of a desire. These mental attitudes respectively represent the information, motivational, and deliberative

states of the agent. The workflow in a generic BDI system is shown in Figure 1 and works as such: a belief revision function receives input information from the environment (e.g., sensors), and it is responsible for updating the belief base. This update can generate more options that can become current desires based on the belief base and the intentions base. A filter is responsible for updating the intentions base, taking into account its previous state and the current belief base and desire base. Finally, an intention is chosen to be carried out as an action by the agent. BDI is the most popular model of agency, it has been and continues to be used in many agent programming languages. AgentSpeak(L) [15] is a language that serves as an abstraction of implemented BDI systems that can be used to interpret agent programs as horn-clause logic programs. The theory behind this language has been implemented as a basis for many APLs.

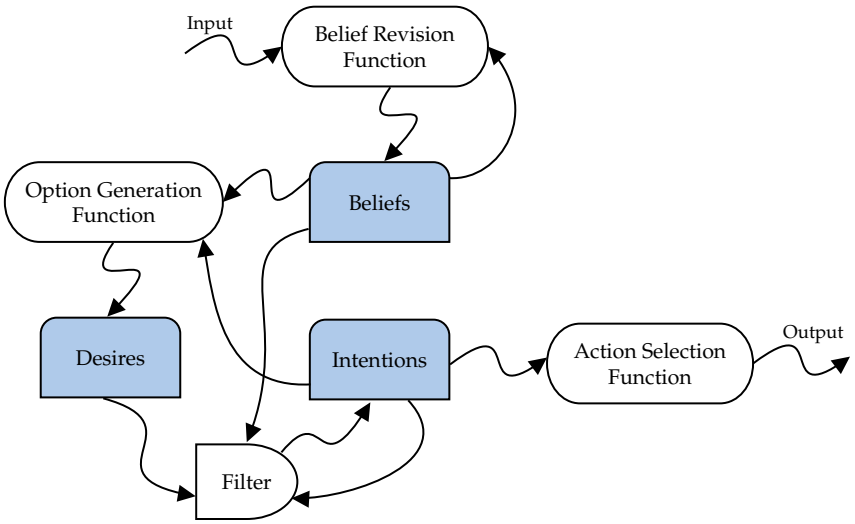

Figure 1. The BDI model.

Situation Calculus [16] is a first order language designed to represent changes in dynamic environments. A situation is a first order term representing a sequence of actions. An initial situation is when no actions have occurred yet. The function $do(a, s)$ results in a successor situation to s after executing the action a, similar to state transition systems. Dynamic environments play an important role in agent-based programming, and as such, Situation Calculus has been used to model how the world changes as a result of executing actions.

As we will see in Section 4, there are many other models that have inspired agent-based programming languages, however, these three were the most influential in the past history of agent-based programming. Some agent languages share similarities or even mix concepts from other programming paradigms, such as procedural, imperative, object-oriented, functional, actor, concurrent, and so on. Comparing the differences or going into detail about these other paradigms is out of scope of this review, but we still consider agent languages that combine concepts from different paradigms.

Historically, agent-based programming has been used in a myriad of practical applications, such as distributed control of electric power grids [17], governance of room allocation [18] and of automated machine-to-machine applications (e.g., traffic redirection) [19], and detecting privacy violations [20]. In Section 4.4, we cover the more recent attempts in using APLs for programming practical applications.

3. Review Methodology

We performed a systematic review of the literature in agent programming languages over the past 5 years (2015–2020). A diagram illustrating our review methodology is shown in Figure 2. For each searched term, we considered the first 10 pages (100 entries) retrieved by Google Scholar (ordered by relevance), for a total of 400 entries (including duplicates).

The terms used in our search were:

- Agent-Based Programming Languages
- Agent-Based Programming Extensions
- Agent-Based Programming Comparison
- Agent-Based Programming Applications

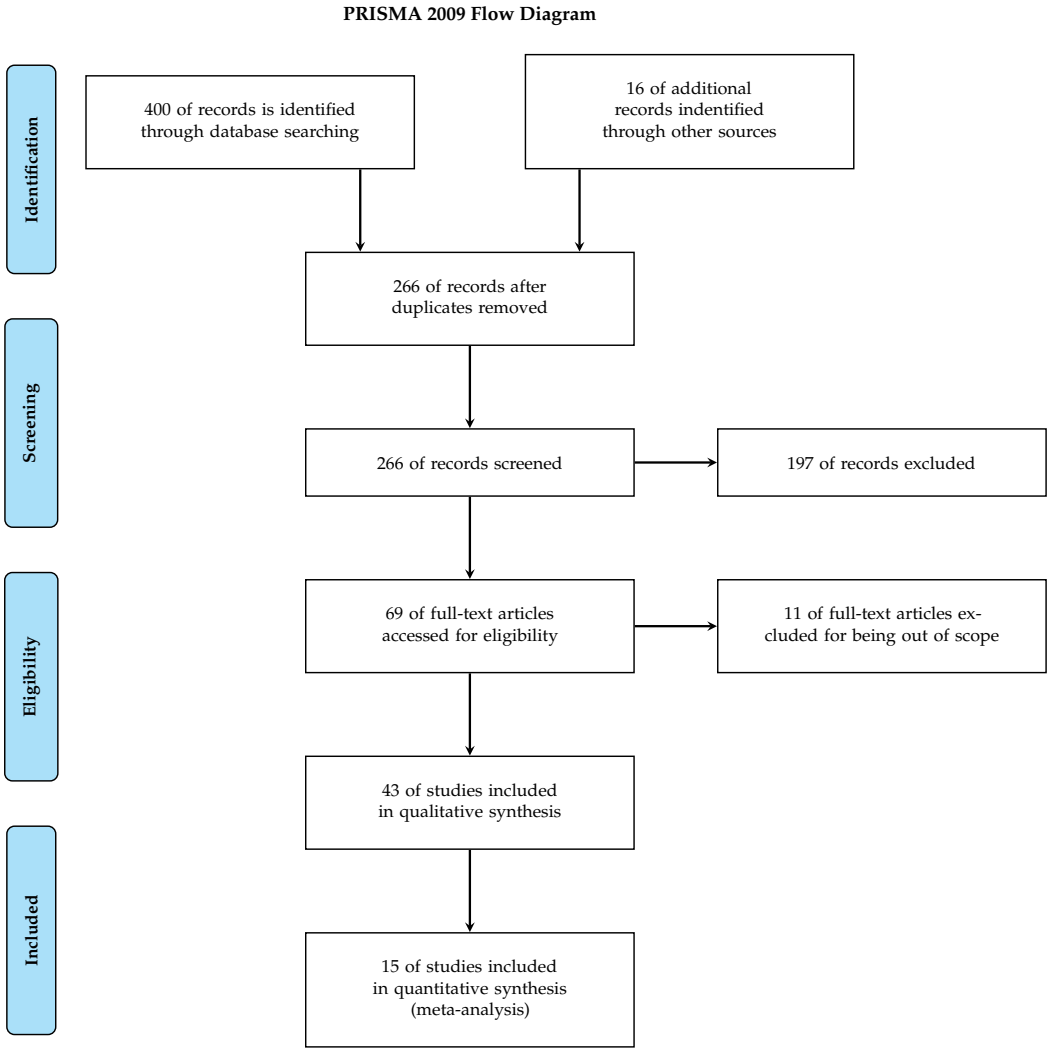

Figure 2. Systematic review flow diagram.

After removing duplicates, we had 250 remaining papers. To these, we added 16 entries from external sources; mostly old references that would not appear in the search, plus a few others found in paper citations and other sources. Out of the 16 external entries, eight were influential APLs that have been developed before 2015 and have been updated recently (i.e., 2017–2020).

4. Review Findings on Agent-Based Programming for Mas

In this section, we cover all of the research found in our systematic review of the literature. We start with the agent programming languages and their extensions, then continue to discuss the existing comparisons in agent-based programming, and close the section with a brief review of applications in the area.

4.1. Agent Programming Languages

We report the agent programming languages found in the systematic review in Table 1. As we alluded to in Section 2, we can see that BDI is clearly the most popular model of

agency, being used in 7 out of the 15 languages. The majority of the implemented languages have been implemented in Java, most likely to take advantage of the cross-platform of Java through its Java Virtual Machine. The table contains only the high-level general-purpose languages that can be used to develop domain independent MASs. While other approaches such as Agent-Based Modelling and Simulation (ABMS) and cognitive agents in robotics are not included in the table, we briefly report some of the novel research that has been done in those areas further below.

Table 1. A collection of recent (or recently updated) agent programming languages. Languages that have no publicly available implementation are represented with ✗. In case there are multiple implementation branches, the Last Updated column refers to the last update in the master branch.

APL	Model	Implementation (Language-Link)	Last Updated
ASTRA	BDI	Java https://gitlab.com/astra-language	6 November 2020
Chromar	rule-based	Haskell https://github.com/azardilis/Chromar	14 June 2020
GOAL	rule-based	Java https://goalapl.atlassian.net/wiki/spaces/GOAL/overview	15 December 2020
Gwendolen	BDI	Java https://github.com/mcapl/mcapl	7 December 2020
JaCaMo	BDI, organisation, environment	Java https://github.com/jacamo-lang/jacamo	20 September 2020
JADE	FIPA	Java https://jade.tilab.com/	8 June 2017
JADEL	DSL, interaction	Java/Jade ✗	✗
Jadex	mixed, BDI and OOP	Java https://github.com/actoron/jadex	10 January 2021
Jadescript	DSL, scripting	Java/Jade ✗	✗
Jason	BDI	Java https://github.com/jason-lang/jason	12 November 2020
LightJason	BDI	Java https://github.com/LightJason/	29 December 2020
PLACE	BDI, HTN	✗	✗
PLASA	Wait-Look-Compute-Move	Java ✗	✗
RMAS	database-centric, CPS	Matlab/SQLite ✗	✗
SARL	DSL	Java https://github.com/sarl/sarl	4 January 2021

4.1.1. General-Purpose APLs

A survey on agent-oriented programming from the software engineering perspective can be found in [6]. One of the main challenges reported in the survey for APL developers is the need to bridge the cognitive gap that exists between the concepts underpinning mainstream languages and those underpinning AOP. In [21] the authors try to fill this gap focusing on understanding the relationship between AgentSpeak(L) [15] and OOP with the goal of trying to reduce the perceived cognitive gap. Such a work proposes a new statically typed agent programming language entitled ASTRA.

In [22] the authors present Chromar, a rule-based notation with stochastic semantics yielding a continuous time Markov chain. Chromar is embedded in Haskell, this gives

it an increased expressive power, and fits with the availability of rich types. In Chromar, rules are first-order abstractions that can both describe a (possibly partial) behaviour of an individual agent and a synchronised action of two or more agents.

GOAL [23] is a declarative agent programming language that uses knowledge base beliefs and goals to support the decision-making of its cognitive agents. Despite sharing similar concepts with the BDI model (beliefs, desires/goals), the GOAL language is more centred towards rule-based decision-making. Agents programs are written in GOAL's specific syntax, but the knowledge of the agent (e.g., rules) are usually represented in Prolog.

Jason [24] is an extension of the AgentSpeak(L) language, based on the BDI agent model. Agents in Jason react to events in the system by executing actions on the environment, according to the plans available in each agent's plan library. One of the extensions in Jason is the addition of Prolog-like rules that can be added and used in the belief base of agents.

The JaCaMo platform [25,26] is composed of three technologies, Jason, CArtAgO [27], and Moise [28], 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 multi-agent systems.

Gwendolen [29] initially started as a small subset of Jason in the hopes of developing verifiable agent programs, but has since grown into its own syntax and semantic. Because it is a language that has been built to support agent verification from the ground up, it is limited in what features it can support, however, the basics of AgentSpeak(L) and BDI are all present. There is a vast literature in verification of agent programs and MAS, but we consider them out of scope for this review. Gwendolen, apart from being verifiable, is still a viable language for developing general-purpose MASs.

JADE [30] is an open source platform for the development of peer-to-peer 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.

In [31] the authors present JADEL (JADE Language), an extension of JADE that provides support for the construction of agents and MAS on top of JADE without having to use Java directly; subsequently, in [32], the authors present Jadescript, an extension of JADEL. Jadescript is characterised by a strong expressive syntax largely inspired by modern scripting languages in order to promote readability and to make agent programs more similar to pseudocode.

Jadex [33] allows the programming of intelligent software agents in XML and Java. The agent abstraction is based on the BDI model, and provides several features such as: a runtime infrastructure for agents, multiple interaction styles, simulation support, automatic overlay network formation, and an extensive runtime tool suite.

LightJason, a highly scalable Java-based platform for BDI agent-oriented programming and simulation is presented in [34]. LightJason is based on a logic language which extends AgentSpeak(L) with lambda-expressions, multi-plan and -rule definition, explicit repair actions, multi-variable assignments, parallel execution, and thread-safe variables. Even though the language is inspired by AgentSpeak(L) and Jason, LightJason is implemented from scratch.

In [35] an AOP language called Planning based Language for Agents and Computational Environments (PLACE) adds AI planning capability to agents. PLACE has a syntactic structure close to BDI, while the planning is done in a Hierarchical Task Network

(HTN) planner. In contrast to other AOP languages, actions in PLACE have durations associated to them, thus, requiring the planner to be able to handle temporal information. Agents in PLACE have the ability to recover from failures by adapting their activities to the new situations. For this purpose, a plan repairing mechanism is added that repairs a plan if the unanticipated events in the environment cause the plan to become unfeasible.

In [36], the Programming Language for Synchronous Agents (PLASA) is proposed. PLASA is platform-independent and facilitates a rapid implementation of co-operative applications on multiple physical robots and in dynamic environments. Essentially, PLASA implements a variant of the Wait-Look-Compute-Move model proposed in [37], where robots move synchronously. It is designed as a high-level programming language, which allows users to specify the instructions to be performed by robots in a human-readable language.

Relational Model Multi-Agent System (RMAS) [38] is a database-centric approach for multi-agent systems suitable for the embodiment of reasoning and control in Cyber-Physical Systems (CPS). Initial implementation of RMAS is proposed by the coupling of the Matlab environment and the SQLite database language.

SARL's [39] focus is to provide an extensible language that is equipped with the minimum amount of concepts (i.e., key concepts) required to support AOP. The language aims to provide abstractions for concurrency, distribution, interaction, decentralisation, reactivity, autonomy, and dynamic reconfiguration. To do so it is not based in any model, but instead it creates its own Domain-Specific Language (DSL) in order to provide a reduced and more lightweight core.

4.1.2. ABMS, Robotics, and Others

As recognised in [40], there is a gap between Agent-Oriented Software Engineering (AOSE) methodologies and the development of ABMS. To overcome this issue, in [41] an AOSE process called Process for Developing Efficient Agent-Based Simulators (PEABS) is proposed. It uses the INGENIAS methodology [42] for modelling the specification and designing its structure. It applies an adaptation framework that allows ABMS developers to obtain simulations with a high efficiency for large amounts of data. Another approach for developing ABMS is presented in [43,44], where the authors propose a new cognitive agent architecture based on the BDI model and integrated into the GAMA modelling language [45]. With respect to previous integration works between BDI and ABMS, in [43] the architecture proposed aims to be flexible and easy to use for non-expert users. Another work which aims to integrate BDI and ABMS is presented in [46], where the authors present a framework that allows BDI cognitive agents to be embedded in an ABMS system. Compared to [43], reference [46] is more general since its objective is to integrate any BDI-based system with ABMS. The only requirement is that the percepts (or environmental observations/events) of interest to each agent and the actions that the agent may execute in the simulation environment can be identified a priori.

ALLEGRO (=ALGOL in PREGO [47,48]) is a programming formalism based on belief architecture for stochastic domains which is intended as an alternative to GOLOG [49] for high-level control in robotic applications. Another language which is based in GOLOG and the situation calculus is introduced in [50], where a prototype implementation of Yet Another GOLOG Interpreter (YAGI), an action-based robot and agent programming language, is presented. YAGI offers bindings for popular robotics frameworks such as Robot Operating System (ROS) [51] and Fawkes.

In [52] the authors present a Cognitive Affective Agent Programming Framework (CAAF), a framework based on the belief-desire theory of emotions that enables the computation of emotions for cognitive agents (i.e., making them cognitive affective agents). The authors present semantics showing the programming constructs of these agents. With these constructs, a programmer can build an agent program with cognitive agents that automatically compute emotions during runs.

4.2. Agent Programming Languages Extensions

In the previous section, we reported works presenting novel APLs (2015–2020), along with works presenting the most influential APLs ($\leq$2015) that are still maintained. Now, we consider the most influential works that have extended existing APLs. We refer to APL extensions as works that have changed existing APLs internally, either by adding new features or by building on top of existing APLs, for example, by customising the APL for a new and specific scenario.

An enhanced version of Multi-Agent System for Competitive Electricity Markets (MASCEM) [53] is presented in [54]. This extended version of the MASCEM simulator aims at supporting the integration of new and complementary models. The facilitation in accommodating different tools and mechanisms is provided by important structural implementation decisions, making MASCEM able to deal with the constantly changing and highly demanding environment of electricity markets. In particular, the new extension of MASCEM brings the use of ontologies to support players' communications.

In [55], the authors present TABSAOND, an extension of PEABS [41]. The main difference between TABSAOND and PEABS is that the former focuses on the design and implementation of the decision-making processes in non deterministic scenarios. In addition, simulators are now deployed as mobile apps and online tools.

In [56] a conservative synchronisation model is proposed for the SARL language and its runtime platform Janus. Since Janus does not make any assumption on the ordering of the events that are exchanged by the agents, it is not possible to use the Janus platform for agent-based simulation involving time without providing the platform with a specific synchronisation mechanism. A model for such a mechanism is described in their extension.

The authors of [57] propose new programming constructs for integrating an advanced yet rule based emotion model, EMIA [58], in line with the 2APL [59] agent language. The combination of both has been carried out by redefining the syntax, semantics and deliberation cycle of 2APL. This combination mainly focuses on event-based emotion generation, and the resulting simulation shows high believability in the emotions expressed by the agent when responding to the real life scenarios.

ARGO [60] is a customised Jason architecture for programming embedded robotic agents using the Javino middleware and perception filters.

In [61], the authors show how procedural reflection in the agent programming language meta-APL [62] can be used to allow a straightforward implementation of some of the steps in the deliberation cycle of a BDI agent, by allowing both agent programs and the agent's deliberation strategy to be encoded in the same programming language.

An extension [63] to Jason and Gwendolen allows the agents in these languages to communicate with ROS, thus supporting the programming of autonomous agents that can control and perform high-level decision-making in robotic applications developed in ROS. The extension is done through an interface that is used as the environment between the agent and ROS, and the communication between the environment and ROS nodes is performed using the *rosbridge* library. The main difference between their work and past attempts at extending traditional APLs to support ROS is that their approach requires no additional modifications in either of the two APLs or ROS, making it usable and portable to different versions of these tools. Similarly, in [64] a framework for using Jason with ROS in embedded systems is presented and a new architecture is introduced to support lower-level interactions between ROS and the agent.

Finally, in [65] a model for a BDI agent programming framework integrating reinforcement learning and an implementation based on the Jason programming language are introduced. The approach supports the design of BDI agents where some plans can be explicitly programmed and others instead can be learned by the agent during the development/engineering stage.

4.3. Agent Programming Languages Comparison

From the research described in the previous two sections we can observe that there are many APLs for developing MAS available in the agent-based programming community. Unfortunately, very often the evaluation of a language is partially or even completely missing. Some studies such as [66] have been done in the past to compare agent languages with other paradigms, in that case the comparison was with actor-based languages. In their results the authors have shown that agent languages (specifically Jason in that work) can indeed be competitive with more lightweight languages such as actors.

In [67], the authors present an evaluation framework for assessing existing or newly defined domain-specific modelling languages for MASs. The evaluation targets both the language and the corresponding tools and provides both qualitative and quantitative results.

A comparison between the pseudocode of a well-known algorithm for solving distributed constraint satisfaction problems and the implementation of such an algorithm in JADEL is shown in [68].

The work in [69] focuses on comparing parallel platforms that support multi-agent simulations and their execution on high performance resources as parallel clusters.

The authors of [70] perform a systematic evaluation of ABMS approaches differentiating the concepts of how complex the model behaviour is and how complicated the model structure is, and illustrate the non-linear relationship between them. Then, they evaluate the trade-offs between simple (often theoretical) models and complicated (often empirically-grounded) models.

4.4. Agent-Based Applications

In this section we list some of the latest applications using agent-based programming. Our goal here is to show the wide variety of application domains that agents can be useful in, thus, this list is not exhaustive and not the main focus of our review.

The Multi-Agent Programming Contest (https://multiagentcontest.org/) (MAPC) is an annual international competition that occurs since 2005. Its purpose is to stimulate research in multi-agent programming by introducing complex benchmark scenarios that require coordinated action and can be used to test and compare multi-agent programming languages, platforms, and tools. Implementations using different agent-based platforms and languages have been used in the last few years; such as JaCaMo [71–73], Jason [74,75], GOAL [76].

Agent-based models to simulate and evaluate the transmission of the coronavirus disease (COVID-19) have been proposed in [77,78]. There is an entire research area focused on using agent-based technologies in the energy industry. For example, in [79], MAS technologies are used for the control of Microgrid, its optimisation and market distribution. For further reading, there is a survey [80] on the applications of MAS in the control and operation of Microgrids, and a review [81] of the state of the art in the application of MAS to energy optimisation problems.

In [82], an application of ABMS is presented to study the relationships between human activities and land-use/land-cover changes to support scientific decisions regarding reasonable land planning and land use. The model is implemented based on the Repast modelling platform [83].

In [84], the authors present and illustrate FlowLogo, an interactive modelling environment for developing coupled agent-based groundwater models. FlowLogo is implemented in NetLogo and is the first integrated software offering a straightforward way to represent agent behaviours that evolve with groundwater conditions. A systematic survey on ABMS tools and applications can be found in [8].

A methodological guide to the use of BDI agents in social simulations and an overview of existing methodologies and tools for using them is provided in [10].

Agents can be used for developing self-managed Internet of Things (IoT) systems due to their distributed nature, context-awareness and self-adaptation (for further reading

about using microservices as agents in IoT [85,86]). In [87] the authors aim to enhance the development of IoT applications using agents and software product lines in self-management systems.

In [88], experiments to validate the programming of autonomous robots using Jadescript are presented. It presents the novel support for perception handlers that has been recently introduced in the language to cope with the high data rate of sensors in robotic applications.

5. Discussion and Future Directions

As we have shown, there exists a wide variety of options for agent-based programming, from more traditional approaches (e.g., BDI) to simulation or planning-based. Some languages have also attempted to combine concepts from agent-based programming with other programming paradigms, most prominently from object-oriented programming. One of the main drawbacks of trying to achieve a wider community of programmers in agent-based programming is the lack of knowledge and familiarity with its concepts, that are significantly different from other more common paradigms. Agent-based programming languages that use some of the concepts from these other paradigms can help bring new programmers that would otherwise be too intimidated. These hybrid languages have their own niche of applications depending on the concepts that they use, which may sometimes overlap with the more "pure" agent programming languages, but "pure" approaches are still necessary to fully tackle all agent programming abstractions (agent, environment, organisation, interaction, etc.).

Out of the 15 agent programming languages listed in Table 1, five do not have publicly available implementation (i.e., the code is not hosted in a public/accessible domain). This represents a significant issue, as it limits the practical usability of the proposed language and makes it difficult to quantitatively compare other languages against it. Not all extensions to existing languages require an implementation to be useful, however, having one available is always positive for the community.

Even though there are several qualitative (e.g., concepts, features) comparisons in the literature, the use of different models of agency makes it difficult to provide a fair comparison between the features present in these languages. A more in-depth study has to be conducted to identify the fundamental features of agent oriented programming, and more importantly, how these features fit in the different models of agency that existing programming languages use.

Quantitative (e.g., performance) comparisons of programming languages are trickier due to the development cycle of having constant updates, which is even more common in programming languages developed in academia (as most of the agent programming languages are). Nevertheless, it is important to develop agent-specific benchmarks that can be used easily by the community to evaluate new programming languages or extensions to existing languages.

Most languages offer a range of different examples that showcase their features and strengths. While these examples are certainly useful to better understand and learn the language, they are usually not enough to convince new users of the applicability of the language in real-world applications. Complex and realistic case studies are hard to develop, but the agent community has available a suite of complex scenarios as part of the annual MAPC that could be better exploited to test and compare agent programming languages.

Two recent surveys [3,5] focused on BDI agent programming outline the limitations and challenges in the area. In a manifesto [3], the author argues that it is necessary to extend the feature set of current APLs to enable wider adoption of agent technology. The author also disagrees with past surveys that the lack of more polished methodologies and tools is not the main factor (although it does contribute to) in the limited adoption of APLs; instead, the author suggests that there is little to no incentive for developers to make the change to AOP, as the behaviours currently shown in applications from the literature can be implemented in more mainstream languages with limited effort. The survey in [5] recaps

the history so far and the state of the art in agent programming with a focus on BDI-based approaches. They identify as a major challenge for future research the integration of AI techniques in agent programming languages as an important and necessary step to the widespread acceptance and adoption of AOP.

Recommendation for Further Research

Considering the past 5 years of research on APLs, many novel frameworks, platforms and models have been proposed. Each one of these, along with new extensions, enriched the agent-based literature and enlarged the spectrum of possible applications. Nonetheless, as rightfully observed in [3,5], the major issue in current APLs is not in their set of features, but in their usability. Generally, there is no desire in learning new languages when the advantages are not straightforward. In our review, we analysed APLs that were both expressive and powerful, but with major usability issues; such as the absence of a (maintained) tool, documentation, and qualitative and quantitative comparisons with other languages. In our opinion, further research on APLs will have to tackle these usability issues in order to try to spread the use of APLs outside the agent community.

6. Conclusions

Agent-based programming is a thriving research area of artificial intelligence. In this review paper, we have classified both veteran and recent contributions according to four different categories: agent-based programming languages, their extensions, the comparisons between languages, and finally, some of the applications using these languages. For each contribution we briefly reviewed the content and outlined the key results. To have a better understanding of the current state of art, we did not only focus on the latest approaches, but we also briefly reviewed the most prominent agent-based programming languages that are still being maintained.

Every year there are many extensions to existing languages and even entire new languages being proposed, however, most of them are limited to formal descriptions without any implementation to support the formal theory. The small subset of approaches with implementation lack any effective evaluation. Comparing new approaches to the state of the art is one of the major steps required to advance the area of agent-based programming. Qualitative and quantitative comparisons can help to identify gaps in existing languages, which can lead to either improvements or new approaches that are able to cope with the challenges raised. Moreover, in our review we have also identified a lack of real-world applications. In order to widen the use of these languages, it is important that their usability in the real-world is well documented, thus, we encourage and recommend more application-based papers that can demonstrate features of agent-based programming in the real-world.

Funding: This research was funded by the UK Industrial Strategy Challenge Fund (ISCF) delivered by UK Research and Innovation (UKRI) and managed by Engineering and Physical Sciences Research Council (EPSRC) under the Robotics and AI for Extreme Environments programme with grants Robotics and AI in Nuclear (RAIN) Hub (EP/R026084/1), Future AI and Robotics for Space (FAIR-SPACE) Hub (EP/R026092/1), and Offshore Robotics for Certification of Assets (ORCA) Hub (EP/R026173/1).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
AOP	Agent-Oriented Programming
AOSE	Agent-Oriented Software Engineering

APL	Agent Programming Language
BDI	Belief-Desire-Intention
CPS	Cyber-Physical Systems
DSL	Domain Specific Language
HTN	Hierarchical Task Network
IoT	Internet of Things
MAPC	Multi-Agent Programming Contest
MAS	Multi-Agent System
OOP	Object-Oriented Programming
PRS	Procedural Reasoning System

References

1. Wooldridge, M. *An Introduction to MultiAgent Systems*, 2nd ed.; John Wiley and Sons: Hoboken, NJ, USA, 2009; ISBN 047149691X.
2. Wooldridge, M.J.; Jennings, N.R. Intelligent agents: theory and practice. *Knowl. Eng. Rev.* **1995**, *10*, 115–152. [CrossRef]
3. Logan, B. An agent programming manifesto. *Int. J. Agent-Oriented Softw. Eng.* **2018**, *6*, 187–210. [CrossRef]
4. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2010.
5. Bordini, R.H.; Seghrouchni, A.E.F.; Hindriks, K.V.; Logan, B.; Ricci, A. Agent programming in the cognitive era. *Auton. Agents Multi Agent Syst.* **2020**, *34*, 37. [CrossRef]
6. Mao, X.; Wang, Q.; Yang, S. A survey of agent-oriented programming from software engineering perspective. *Web Intell.* **2017**, *15*, 143–163. [CrossRef]
7. Kravari, K.; Bassiliades, N. A Survey of Agent Platforms. *J. Artif. Soc. Soc. Simul.* **2015**, *18*. [CrossRef]
8. Abar, S.; Theodoropoulos, G.K.; Lemarinier, P.; O'Hare, G.M.P. Agent Based Modelling and Simulation tools: A review of the state-of-art software. *Comput. Sci. Rev.* **2017**, *24*, 13–33. [CrossRef]
9. Isern, D.; Moreno, A. A systematic literature review of agents applied in healthcare. *J. Med Syst.* **2016**, *40*, 43. [CrossRef]
10. Adam, C.; Gaudou, B. BDI agents in social simulations: a survey. *Knowl. Eng. Rev.* **2016**, *31*, 207–238. [CrossRef]
11. Shoham, Y. Agent-oriented Programming. *Artif. Intell.* **1993**, *60*, 51–92. [CrossRef]
12. Georgeff, M.; Lansky, A. Procedural Knowledge. *Proc. IEEE (Spec. Issue Knowl. Represent.)* **1986**, *74*, 1383–1398. [CrossRef]
13. Bratman, M.E. *Intentions, Plans, and Practical Reason*; Center for the Study of Language and Information: Stanford, CA, USA, 1999.
14. Rao, A.S.; Georgeff, M. BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multiagent Systems (ICMAS), San Francisco, CA, USA, 12–14 June 1995; pp. 312–319.
15. Rao, A.S. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In *Agents Breaking Away, Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, 22–25 January 1996*; Lecture Notes in Computer Science; de Velde, W.V., Perram, J.W., Eds.; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1038, pp. 42–55. [CrossRef]
16. McCarthy, J.; Hayes, P.J. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence 4*; Meltzer, B., Michie, D., Eds.; Edinburgh University Press: Edinburgh, UK, 1969; pp. 463–502. reprinted in McC90.
17. Issicaba, D.; Rosa, M.A.; Prostejovsky, A.M.; Bindner, H.W. Experimental validation of BDI agents for distributed control of electric power grids. In Proceedings of the 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Torino, Italy, 26–29 September 2017; pp. 1–6. [CrossRef]
18. Sorici, A.; Boissier, O.; Picard, G.; Santi, A. Exploiting the JaCaMo Framework for Realising an Adaptive Room Governance Application. In Proceedings of the Compilation of the Co-Located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, and VMIL'11; New York, NY, USA, 1–31 October 2011; pp. 239–242. [CrossRef]
19. Persson, C.; Picard, G.; Ramparany, F.; Boissier, O. A JaCaMo-Based Governance of Machine-to-Machine Systems. In *Advances on Practical Applications of Agents and Multi-Agent Systems*; Demazeau, Y., Müller, J.P., Rodríguez, J.M.C., Pérez, J.B., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 161–168.
20. Krupa, Y.; Vercouter, L. Handling Privacy as Contextual Integrity in Decentralized Virtual Communities: The PrivaCIAS Framework. *Web Intelli. Agent Syst.* **2012**, *10*, 105–116. [CrossRef]
21. Collier, R.W.; Russell, S.E.; Lillis, D. Reflecting on Agent Programming with AgentSpeak(L). In *Proceedings of the PRIMA 2015: Principles and Practice of Multi-Agent Systems—18th International Conference, Bertinoro, Italy, 26–30 October 2015*; Lecture Notes in Computer Science; Chen, Q., Torroni, P., Villata, S., Hsu, J.Y., Omicini, A., Eds.; Springer: Cham, Switzerland, 2015; Volume 9387, pp. 351–366. [CrossRef]
22. Honorato-Zimmer, R.; Millar, A.J.; Plotkin, G.D.; Zardilis, A. Chromar, a language of parameterised agents. *Theor. Comput. Sci.* **2019**, *765*, 97–119. [CrossRef]
23. Hindriks, K.V.; de Boer, F.S.; van der Hoek, W.; Meyer, J.J.C. Agent Programming with Declarative Goals. In Proceedings of the 7th International Workshop on Agent Theories, Architectures, Boston, MA, USA, 7–9 July 2020; pp. 228–243.
24. Bordini, R.H.; Wooldridge, M.; Hübner, J.F. *Programming Multi-Agent Systems in AgentSpeak Using Jason*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
25. Boissier, O.; Bordini, R.H.; Hübner, J.F.; Ricci, A.; Santi, A. Multi-agent oriented programming with JaCaMo. *Sci. Comput. Program.* **2013**, *78*, 747–761. [CrossRef]

26. 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: Cambridge, MA, USA, 2020.
27. Ricci, A.; Piunti, M.; Viroli, M.; Omicini, A. Environment Programming in CArtAgO. In *Multi-Agent Programming: Languages, Tools and Applications*; Multiagent Systems, Artificial Societies, and Simulated Organizations; Springer: Boston, MA, USA, 2009; Chapter 8, pp. 259–288. [CrossRef]
28. 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.* **2007**, *1*, 370–395. [CrossRef]
29. Dennis, L.A. *Gwendolen Semantics: 2017*; Technical Report ULCS-17-001; University of Liverpool, Department of Computer Science: Liverpool, UK, 2017.
30. Bellifemine, F.L.; Caire, G.; Greenwood, D. *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
31. Bergenti, F.; Iotti, E.; Monica, S.; Poggi, A. Agent-oriented model-driven development for JADE with the JADEL programming language. *Comput. Lang. Syst. Struct.* **2017**, *50*, 142–158. [CrossRef]
32. Bergenti, F.; Monica, S.; Petrosino, G. A scripting language for practical agent-oriented programming. In Proceedings of the 8th ACM SIGPLAN International Workshop on Programming Based on Actors, Agents, and Decentralized Control, AGERE!@SPLASH 2018, Boston, MA, USA, 5 November 2018; pp. 62–71. [CrossRef]
33. Pokahr, A.; Braubach, L.; Lamersdorf, W., Jadex: A BDI Reasoning Engine. In *Multi-Agent Programming: Languages, Platforms and Applications*; Springer: Boston, MA, USA, 2005; pp. 149–174. [CrossRef]
34. Aschermann, M.; Dennisen, S.; Kraus, P.; Müller, J.P. LightJason, a Highly Scalable and Concurrent Agent Framework: Overview and Application. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, 10–15 July 2018; pp. 1794–1796.
35. Hashmi, M.A.; Seghrouchni, A.E.F.; Akram, M.U. A Planning Based Agent Programming Language Supporting Environment Modeling. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2015, Singapore, 6–9 December 2015; pp. 76–83. [CrossRef]
36. Kilaru, J. PLASA: Programming Language for Synchronous Agents. Master's Thesis, California State University, Long Beach, CA, USA, 2018.
37. Flocchini, P.; Prencipe, G.; Santoro, N.; Widmayer, P. Gathering of asynchronous robots with limited visibility. *Theor. Comput. Sci.* **2005**, *337*, 147–168. [CrossRef]
38. Bonci, A.; Pirani, M.; Bianconi, C.; Longhi, S. RMAS: Relational Multiagent System for CPS Prototyping and Programming. In Proceedings of the 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, MESA 2018, Oulu, Finland, 2–4 July 2018; pp. 1–6. [CrossRef]
39. Rodriguez, S.; Gaud, N.; Galland, S. SARL: A General-Purpose Agent-Oriented Programming Language. In Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, Poland, 11–14 August 2014; Volume III, pp. 103–110; [CrossRef]
40. Molesini, A.; Casadei, M.; Omicini, A.; Viroli, M. Simulation in agent-oriented software engineering: The SODA case study. *Sci. Comput. Program.* **2013**, *78*, 705–714. [CrossRef]
41. García-Magariño, I.; Gómez-Rodríguez, A.; Moreno, J.C.G.; Navarro, G.P. PEABS: A Process for developing Efficient Agent-Based Simulators. *Eng. Appl. Artif. Intell.* **2015**, *46*, 104–112. [CrossRef]
42. Pavón, J.; Gómez-Sanz, J.; Fuentes-Fernández, R., The INGENIAS methodology and tools. In *Agent-Oriented Methodol*; IGI Global: Hershey, PA, USA, 2005; pp. 236–276. [CrossRef]
43. Caillou, P.; Gaudou, B.; Grignard, A.; Truong, Q.C.; Taillandier, P. A Simple-to-Use BDI Architecture for Agent-Based Modeling and Simulation. In Proceedings of the European Social Simulation Association 2015, Groningen, The Netherlands, 14–18 September 2015; Volume 528, pp. 15–28. [CrossRef]
44. Taillandier, P.; Bourgais, M.; Caillou, P.; Adam, C.; Gaudou, B. A BDI Agent Architecture for the GAMA Modeling and Simulation Platform. In Proceedings of the Multi-Agent Based Simulation XVII—International Workshop, MABS 2016, Singapore, 10 May 2016; Volume 10399, pp. 3–23. [CrossRef]
45. Grignard, A.; Taillandier, P.; Gaudou, B.; Vo, D.; Huynh, N.Q.; Drogoul, A. GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation. In Proceedings of the PRIMA 2013: Principles and Practice of Multi-Agent Systems—16th International Conference, Dunedin, New Zealand, 1–6 December 2013; Volume 8291, pp. 117–131. [CrossRef]
46. Singh, D.; Padgham, L.; Logan, B. Integrating BDI Agents with Agent-Based Simulation Platforms. *Auton. Agents Multi Agent Syst.* **2016**, *30*, 1050–1071. [CrossRef]
47. Belle, V.; Levesque, H.J. PREGO: An Action Language for Belief-Based Cognitive Robotics in Continuous Domains. In Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014; pp. 989–995.
48. Belle, V.; Levesque, H.J. ALLEGRO: Belief-Based Programming in Stochastic Dynamical Domains. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, 25–31 July 2015; pp. 2762–2769.
49. Levesque, H.J.; Reiter, R.; Lespérance, Y.; Lin, F.; Scherl, R.B. GOLOG: A Logic Programming Language for Dynamic Domains. *J. Log. Program.* **1997**, *31*, 59–83. [CrossRef]

50. Ferrein, A.; Maier, C.; Mühlbacher, C.; Niemueller, T.; Steinbauer, G.; Vassos, S. Controlling Logistics Robots with the Action-Based Language YAGI. In Proceedings of the Intelligent Robotics and Applications—9th International Conference, ICIRA 2016, Tokyo, Japan, 22–24 August 2016; Volume 9834, pp. 525–537. [CrossRef]
51. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A. ROS: An open-source Robot Operating System. In Proceedings of the Workshop on Open Source Software at the International Conference on Robotics and Automation, Kobe, Japan, 12–13 May 2009; p. 5.
52. Kaptein, F.; Broekens, J.; Hindriks, K.V.; Neerincx, M.A. CAAF: A Cognitive Affective Agent Programming Framework. In Proceedings of the Intelligent Virtual Agents—16th International Conference, IVA 2016, Los Angeles, CA, USA, 20–23 September 2016; Volume 10011, pp. 317–330. [CrossRef]
53. Praça, I.; Ramos, C.; Vale, Z.; Cordeiro, M. MASCEM: A multiagent system that simulates competitive electricity markets. *IEEE Intell. Syst.* **2003**, *18*, 54–60. [CrossRef]
54. Santos, G.; Pinto, T.; Praça, I.; Vale, Z. MASCEM: Optimizing the performance of a multi-agent system. *Energy* **2016**, *111*, 513–524. [CrossRef]
55. García-Magariño, I.; Navarro, G.P.; Lacuesta, R. TABSAOND: A technique for developing agent-based simulation apps and online tools with nondeterministic decisions. *Simul. Model. Pract. Theory* **2017**, *77*, 84–107. [CrossRef]
56. Cich, G.; Galland, S.; Knapen, L.; Yasar, A.; Bellemans, T.; Janssens, D. Addressing the Challenges of Conservative Event Synchronization for the SARL Agent-Programming Language. In Proceedings of the Advances in Practical Applications of Cyber-Physical Multi-Agent Systems, PAAMS Collection—15th International Conference, PAAMS 2017, Porto, Portugal, 21–23 June 2017; Volume 10349, pp. 31–42. [CrossRef]
57. Jain, S.; Asawa, K. Programming an expressive autonomous agent. *Expert Syst. Appl.* **2016**, *43*, 131–141. [CrossRef]
58. Jain, S.; Asawa, K. EMIA: emotion model for intelligent agent. *J. Intell. Syst.* **2015**, *24*, 449–465. [CrossRef]
59. Dastani, M. 2APL: A practical agent programming language. *Auton. Agents Multi-Agent Syst.* **2008**, *16*, 214–248. [CrossRef]
60. Pantoja, C.E.; Stabile, M.F.; Lazarin, N.M.; Sichman, J.S. ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In Proceedings of the Engineering Multi-Agent Systems—4th International Workshop, EMAS 2016, Singapore, 9–10 May 2016; Volume 10093, pp. 136–155. [CrossRef]
61. Leask, S.; Logan, B. Programming deliberation strategies in meta-APL. In Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems, Bertinoro, Italy, 26–30 October 2015; pp. 433–448.
62. Doan, T.T.; Yao, Y.; Alechina, N.; Logan, B. Verifying heterogeneous multi-agent programs. In Proceedings of the International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, 5–9 May 2014; pp. 149–156.
63. Cardoso, R.C.; Ferrando, A.; Dennis, L.A.; Fisher, M. An Interface for Programming Verifiable Autonomous Agents in ROS. In Proceedings of the European Conference on Multi-Agent Systems (EUMAS), Thessaloniki, Greece, 14–15 September 2020.
64. Onyedinma, C.; Gavigan, P.; Esfandiari, B. Toward Campus Mail Delivery Using BDI. *J. Sens. Actuator Netw.* **2020**, *9*, 56. [CrossRef]
65. Bosello, M.; Ricci, A. From Programming Agents to Educating Agents - A Jason-Based Framework for Integrating Learning in the Development of Cognitive Agents. In Proceedings of the Engineering Multi-Agent Systems—7th International Workshop, EMAS 2019, Montreal, QC, Canada, 13–14 May 2019; Volume 12058, pp. 175–194. [CrossRef]
66. Cardoso, R.C.; Zatelli, M.R.; Hübner, J.F.; Bordini, R.H. Towards Benchmarking Actor- and Agent-Based Programming Languages. In Proceedings of the Workshop on Programming Based on Actors, Agents, and Decentralized Control, Indianapolis, IN, USA, 27 October 2013; pp. 115–126.
67. Challenger, M.; Kardas, G.; Tekinerdogan, B. A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems. *Softw. Qual. J.* **2016**, *24*, 755–795. [CrossRef]
68. Bergenti, F.; Iotti, E.; Monica, S.; Poggi, A. A Comparison between Asynchronous Backtracking Pseudocode and its JADEL Implementation. In Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART, Porto, Portugal, 24–26 February 2017; Volume 2, pp. 250–258. [CrossRef]
69. Rousset, A.; Herrmann, B.; Lang, C.; Philippe, L. A survey on parallel and distributed multi-agent systems for high performance computing simulations. *Comput. Sci. Rev.* **2016**, *22*, 27–46. [CrossRef]
70. Sun, Z.; Lorscheid, I.; Millington, J.D.A.; Lauf, S.; Magliocca, N.R.; Groeneveld, J.; Balbi, S.; Nolzen, H.; Müller, B.; Schulze, J.; et al. Simple or complicated agent-based models? A complicated issue. *Environ. Model. Softw.* **2016**, *86*, 56–67. [CrossRef]
71. Cardoso, R.C.; Krausburg, T.; Baségio, T.L.; Engelmann, D.C.; Hübner, J.F.; Bordini, R.H. SMART-JaCaMo: An organization-based team for the multi-agent programming contest. *Ann. Math. Artif. Intell.* **2018**, *84*, 75–93. [CrossRef]
72. Krausburg, T.; Cardoso, R.C.; Damasio, J.; Peres, V.; Farias, G.P.; Engelmann, D.C.; Hübner, J.F.; Bordini, R.H. SMART–JaCaMo: An Organisation-Based Team for the Multi-Agent Programming Contest. In *The Multi-Agent Programming Contest 2018*; Ahlbrecht, T., Dix, J., Fiekas, N., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 72–100.
73. Cardoso, R.C.; Ferrando, A.; Papacchini, F. LFC: Combining Autonomous Agents and Automated Planning in the Multi-Agent Programming Contest. In *Multi-Agent Progamming Contest*; Springer: Cham, Switzerland, 2019; pp. 31–58.
74. Vezina, M.; Esfandiari, B. The Requirement Gatherers' Approach to the 2019 Multi-Agent Programming Contest Scenario. In *The Multi-Agent Programming Contest 2019*; Ahlbrecht, T., Dix, J., Fiekas, N., Krausburg, T., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 106–150.

75. Villadsen, J.; Bjørn, M.O.; From, A.H.; Henney, T.S.; Larsen, J.B. Multi-Agent Programming Contest 2018—The Jason-DTU Team. In *The Multi-Agent Programming Contest 2018*; Ahlbrecht, T., Dix, J., Fiekas, N., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 41–71.
76. Jensen, A.B.; Villadsen, J. GOAL-DTU: Development of Distributed Intelligence for the Multi-Agent Programming Contest. In *The Multi-Agent Programming Contest 2019*; Ahlbrecht, T., Dix, J., Fiekas, N., Krausburg, T., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 79–105.
77. Wolfram, C. An Agent-Based Model of COVID-19. *Complex Syst.* **2020**, *29*. [CrossRef]
78. Prudhomme, C.; Cruz, C.; Cherifi, H. An Agent based model for the transmission and control of the COVID-19 in Dijon (extended abstract). In Proceedings of MARAMI 2020—Modèles & Analyse des Réseaux: Approches Mathématiques & Informatiques—The 11th Conference on Network Modeling and Analysis, Virtual Conference, Montpellier, France, 14–15 October 2020; Volume 2750.
79. Khan, M.W.; Wang, J. The research on multi-agent system for microgrid control and optimization. *Renew. Sustain. Energy Rev.* **2017**, *80*, 1399–1411. [CrossRef]
80. Kantamneni, A.; Brown, L.E.; Parker, G.G.; Weaver, W.W. Survey of multi-agent systems for microgrid control. *Eng. Appl. Artif. Intell.* **2015**, *45*, 192–203. [CrossRef]
81. González-Briones, A.; De La Prieta, F.; Mohamad, M.S.; Omatu, S.; Corchado, J.M. Multi-agent systems applications in energy optimization problems: A state-of-the-art review. *Energies* **2018**, *11*, 1928. [CrossRef]
82. QuanLi, X.; Kun, Y.; GuiLin, W.; YuLian, Y. Agent-based modeling and simulations of land-use and land-cover change according to ant colony optimization: A case study of the Erhai Lake Basin, China. *Nat. Hazards* **2015**, *75*, 95–118. [CrossRef]
83. North, M.; Collier, N.; Ozik, J.; Tatara, E.; Macal, C.; Bragen, M.; Sydelko, P. Complex Adaptive Systems Modeling with Repast Simphony. *Complex Adapt. Syst. Model.* **2013**, *1*, 1–26. [CrossRef]
84. Castilla-Rho, J.C.; Mariethoz, G.; Rojas-Mujica, R.; Andersen, M.S.; Kelly, B.F.J. An agent-based platform for simulating complex human-aquifer interactions in managed groundwater systems. *Environ. Model. Softw.* **2015**, *73*, 305–323. [CrossRef]
85. Savaglio, C.; Fortino, G.; Ganzha, M.; Paprzycki, M.; Badica, C.; Ivanovic, M. Agent-Based Computing in the Internet of Things: A Survey. In Proceedings of the Intelligent Distributed Computing XI—11th International Symposium on Intelligent Distributed Computing—IDC 2017, Belgrade, Serbia, 11–13 October 2017; Volume 737, pp. 307–320. [CrossRef]
86. Krivic, P.; Skocir, P.; Kusek, M.; Jezic, G. Microservices as Agents in IoT Systems. In Proceedings of the Agent and Multi-Agent Systems: Technology and Applications, 11th KES International Conference, KES-AMSTA 2017, Vilamoura, Algarve, Portugal, 21–23 June 2017; Volume 74, pp. 22–31. [CrossRef]
87. Ayala, I.; Amor, M.; Fuentes, L.; Troya, J.M. A Software Product Line Process to Develop Agents for the IoT. *Sensors* **2015**, *15*, 15640–15660. [CrossRef]
88. Iotti, E.; Petrosino, G.; Monica, S.; Bergenti, F. Exploratory Experiments on Programming Autonomous Robots in Jadescript. In Proceedings of the First Workshop on Agents and Robots for reliable Engineered Autonomy, AREA@ECAI 2020, Virtual Event, Santiago de Compostela, Spain, 4 September 2020; Volume 319, pp. 55–67. [CrossRef]