Engineering Multi-Agent Systems: State of Affairs and the Road Ahead

Viviana Mascardi[1], Danny Weyns[2], Alessandro Ricci[3]
(Workshop co-chairs)

Clara Benac Earle[4], Arthur Casals[5,10], Moharram Challenger[6], Amit Chopra[7], Andrei Ciortea[8], Louise A. Dennis[9], Álvaro Fernández Díaz[4], Amal El Fallah-Seghrouchni[10], Angelo Ferrando[11], Lars-Åke Fredlund[4], Eleonora Giunchiglia[12], Zahia Guessoum[10], Akin Günay[7], Koen Hindriks[13], Carlos A. Iglesias[4], Brian Logan[14], Timotheus Kampik[15], Geylani Kardas[6], Vincent J. Koeman[13], John Bruntse Larsen[16], Simon Mayer[17], Tasio Méndez[4], Juan Carlos Nieves[15], Valeria Seidita[18], Baris Tekin Tezel[6], László Z. Varga[19], Michael Winikoff[20]
(Contributing workshop participants)

[1] University of Genova, Italy, viviana.mascardi@unige.it
[2] KU Leuven Belgium and Linnaeus University, Sweden, danny.weyns@gmail.com
[3]University of Bologna, Italy, a.ricci@unibo.it

[4]Universidad Politécnica de Madrid, ES, [5]Universidade de São Paulo, BR, [6]Ege University, TR, [7]University of Lancaster, UK, [8]MINES Saint-Étienne, FR, [9]University of Liverpool, UK, [10]Sorbonne Université, FR, [11]University of Genova, IT, [12]University of Oxford, UK, [13]Delft University of Technology, NL, [14]University of Nottingham, UK, [15]Umeå University, SE, [16]Technical University of Denmark, DK, [17]University of St. Gallen and ETH Zurich, CH, [18]Università degli Studi di Palermo and CNR, IT, [19]Eötvös Loránd University, HU, [20]Otago University, NZ

ABSTRACT
The continuous integration of software-intensive systems together with the ever-increasing computing power offer a breeding ground for intelligent agents and multi-agent systems (MAS) more than ever before. Over the past two decades, a wide variety of languages, models, techniques and methodologies have been proposed to engineer agents and MAS. Despite this substantial body of knowledge and expertise, the systematic engineering of large-scale and open MAS still poses many challenges. Researchers and engineers still face fundamental questions regarding theories, architectures, languages, processes, and platforms for designing, implementing, running, maintaining, and evolving MAS. This paper reports on the results of the 6th International Workshop on Engineering Multi-Agent Systems (EMAS 2018, 14th-15th of July, 2018, Stockholm, Sweden), where participants discussed the issues above focusing on the state of affairs and the road ahead for researchers and engineers in this area.

Categories and Subject Descriptors
I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents, Multiagent systems
D.2 [**Software Engineering**]: General

General Terms
Algorithms, Design, Reliability, Experimentation, Security, Human Factors, Standardization, Languages, Theory, Verification.

Keywords
Software engineering, Agents, Multi-Agent Systems, Goal Reasoning, AI.

1. INTRODUCTION

According to many scientists, one way to describe Artificial Intelligence (AI) is as the study of agents that receive percepts from, plan and perform actions in the environment. The main unifying theme underlying AI is then the idea of an intelligent agent able to reason, plan, act, interact, and learn [38]. This metaphor makes intelligent agents appealing for a wide audience interested in classical and distributed AI, agents and multi-agent systems (MAS) engineering, machine learning, and decentralized systems. Despite the substantial body of knowledge and expertise developed in the design and development of MAS, the systematic engineering of large-scale and open MAS still poses many challenges. Even though various languages, models, techniques and methodologies have been proposed, researchers and developers still face fundamental questions attaining the engineering of MAS, and more in detail foundational theories, architectures, languages, processes, and platforms for designing, implementing, running, maintaining and evolving MAS.

In this context, interesting questions to be addressed are:

- How to express the requirements for large-scale and open MAS and how to translate these requirements into agent goals?
- Which architectures are most suitable for MAS of different domains?
- How to seamlessly integrate AI and machine learning techniques into design/programming languages and tools for agent-based systems?
- How to specify, design, implement, verify, test, validate and evolve MAS?
- How to enable agent-based systems to deal with continuous change, for example in the operating environment or user requirements?
- How to ensure/control global behavior of decentralized MAS?
- How to seamlessly integrate MAS engineering with mainstream engineering models, languages, frameworks and tools?
- What are the implications of MAS engineering in the context of continuous development and deployment?
- What is the synergy between Cloud and Edge computing on the one hand and MAS engineering on the other hand?

• How to scale with the complexity of real-world application domains?

• How can MAS help developing Cyber-Physical Systems and Internet of Things (IoT)?

• Which tools and frameworks are available/needed?

• Which processes are required for fast but high-quality development of MAS?

EMAS 2018 zoomed in into several of these questions through an invited talk, focused presentations, panels, and discussion groups. After a short presentation of the workshop and its history, the remainder of the paper summarizes these activities and the main outcomes of the event.

2. EMAS WORKSHOP

EMAS 2018 took place in Stockholm, Sweden, on 14th-15th of July, 2018. It aimed to gather researchers and engineers with an interest in software engineering (SE) and programming of MAS, declarative agent languages and technologies, machine learning, and other AI-related topics to present and discuss their research and emerging results in MAS engineering. The overall purpose of this workshop was to facilitate the cross-fertilization of ideas and experiences in the various fields to:

1. enhance our knowledge and expertise in MAS engineering and improve the state-of-the-art;
2. define new directions for MAS engineering that are useful to practitioners, relying in results and recommendations coming from different but continuous research areas;
3. investigate how practitioners can use or adapt established processes and methodologies for the engineering of large-scale and open MAS;
4. involve more master and PhD students.

The EMAS workshop has been held as part of AAMAS since 2013 and was affiliated to AAMAS through the AOSE, ProMAS and DALT workshops since its inception. The sixth edition of the workshop, co-located for the first time with IJCAI/ECAI and ICML besides AAMAS, followed the successful editions that were held in 2013 in St. Paul, Minnesota, in 2014 in Paris, France, in 2015 in Istanbul, Turkey, in 2016 in Singapore, and in 2017 in São Paulo, Brazil.

The post-proceedings of EMAS 2013 (LNAI 8245), EMAS 2014 (LNAI 8758), EMAS 2015 (LNAI 9318), EMAS 2016 (LNAI 10093), EMAS 2017 (LNAI 10739) have been published by Springer in the Lecture Notes in Artificial Intelligence series. A few special issues of the International Journal of Agent-Oriented Software Engineering[1] arising from EMAS have also been published.

The publication of the EMAS 2018 post-proceedings with Springer in a book under the Lecture Notes in Artificial Intelligence (LNAI) is under way.

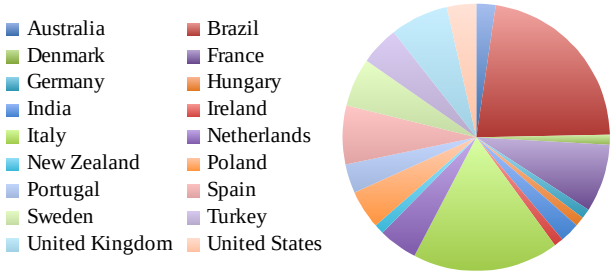

Figure 1: EMAS 2018 submitted papers' authors.

EMAS 2018 received 32 submissions by authors from all around the world, as shown in Figure 1. From 32 submissions, 21 papers were accepted in the following categories:

- 11 of 22 regular paper submissions accepted
- 5 accepted as short paper
- 2 of 4 short paper submissions accepted
- 1 of 2 demo submissions accepted
- 0 of 1 extended abstracts accepted
- 2 of 3 doctoral project submissions accepted

EMAS 2018 program committee consisted of 41 scientists from 16 different countries (Figure 2).

More than 50 persons attended the events scheduled in the first day, and about 30 attended the second day.

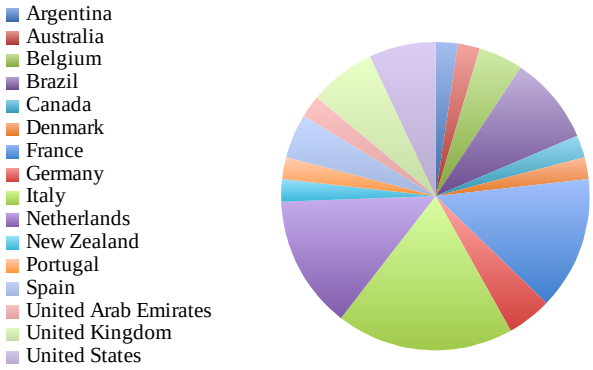

Figure 2: EMAS 2018 PC members.

3. WORKSHOP INVITED TALK

The invited talk was given by Simon Mayer, now with the University of St. Gallen as a Professor of Interaction- and Communication-based Systems, and dealt with Autonomous Agents for Flexible Hypermedia Systems.

The Web of Things (WoT) community used to be driven by the application of Web technologies to enable flexible mash-ups of smart devices on top of the Internet of Things, an objective that we consider accomplished (from a research standpoint) in many different domains ranging from smart homes and cars to dynamic factories in the Industry 4.0 paradigm. However, work in the WoT space on engineering interacting systems of smart devices - “physical mash-ups” - is tightly connected to work in the Semantic Web community and in the AAMAS domain: one of the next big things for the WoT community is to increase the autonomy of Web-enabled devices and their understanding of one another. This can be done, for example, by outfitting the devices with semantic descriptions of their properties and functions and, sometimes, even bestowing agency upon them. In the talk, Mayer discussed this convergence that will enrich real-world devices with AAMAS technologies, and open up real-world applications to the AAMAS community, while examining important properties of the Web architecture that support flexibly interacting autonomous things on the Web. He placed a particular emphasis on the HATEOAS principle or REST: *hypermedia as the engine of application state*, which directly supports the creation of *local* mash-ups (i.e. mash-ups of services that are hosted on a single server, or a conglomerate of friendly servers with mutual interlinking). While HATEOAS is thus sufficient for some use cases, it usually needs to be paired with a mechanism to enable *global* service mash-ups, such as AI planning [25] potentially in combination with techniques from the AAMAS and EMAS space [11].

[1] http://www.inderscience.com/jhome.php?jcode=ijaose

4. WORKSHOP TECHNICAL PRESENTATIONS

The workshop presentations focused on the following themes: programming agents and MAS, Agent-Oriented Software Engineering (AOSE), formal analysis & techniques, rational agents techniques, modeling & simulations, frameworks and application domains.

4.1 Programming agents and MAS

The paper "*Pitfalls of Jason Concurrency*" by Álvaro Fernández Díaz, Clara Benac Earle and Lars-Åke Fredlund examines to what extent the Jason programming language [6] aids programmers in coping with the difficulties caused by intra-agent concurrency, e.g., race conditions due to the presence of multiple agent intentions. Roughly, such difficulties can be classified as either being caused by (i) unexpected interference from concurrent computations, or (ii) due to the unexpected timing of events. The paper analyses a number strategies to mitigate concurrency problems present either in the original Jason language, or in later language extensions. Such mitigations are often realized as Jason implementation options, instead of realizing changes to the underlying Jason semantics. Relying on such optional behaviors carries the risk that the behavior of a Jason program cannot be understood by examining its source code alone.

Alessandro Ricci, Rafael H. Bordini, Jomi F. Hubner and Rem Collier present "*AgentSpeak(ER): Enhanced Encapsulation in Agent Plans*". AgentSpeak(ER) extends AgentSpeak(L) [36] to support encapsulation and allows for improving the style of Belief-Desire-Intentions (BDI) agent programming along relevant aspects, including program modularity and readability, failure handling, and reactive as well as goal-based reasoning. AgentSpeak(ER) has been implemented and experimentally evaluated on top of the ASTRA [13] platform and an implementation in Jason is under way.

4.2 Agent-Oriented Software Engineering

In their paper "*Improving the Usability of a MAS DSML*", which received the **best paper award**, Tomás Miranda, Moharram Challenger, Baris Tezel, Omer Faruk Alaca, Vasco Amaral, Miguel Goulão and Geylani Kardas point out the need for evaluating the usability of domain-specific modeling languages (DSMLs) for MASs especially to leverage the adoption of such languages by the agent developers during MAS design and implementation. Many MAS DSMLs are proposed by the AOSE researchers along with appropriate IDEs in which both modeling and code generation can be performed [21]. However, the evaluation of these DSMLs is completely missing or has been carried out with an idiosyncratic approach [9]. Miranda et al. focus on the usability of DSMLs for MAS and introduce an approach for promoting the usability of such languages by applying the principles of the "Physics" of Notations (PoN) [27]. For this purpose, the visual notation of a MAS DSML, called SEA_ML [8], was evaluated and modified according to PoN principles which led to the development of the new version of the language, called SEA ML++. SEA ML++ was perceived as significantly improved in terms of the concrete syntax's comprehensibility, adequacy and usability, as a direct result of employing the PoN principles.

Artur Freitas, Rafael H. Bordini and Renata Vieira present their proposal for the "*Automatic Generation of Multi-Agent Programs from Ontology Models*". The foundation of such work, aimed at facilitating the development of MASs designed as ontology models supporting code generation, takes into consideration ontologies for agent-oriented software engineering aligned with the JaCaMo framework [5]. These techniques are implemented in a tool that supports multi-agent systems core code generation for JaCaMo, and the underlying ontology also allows for reasoning about the MAS models under development.

Massimo Cossentino, Luca Sabatucci and Valeria Seidita discuss the "*Lesson Learnt from Designing Self-Adaptive Systems with MUSA*" and deal with complex-self adaptive systems operating in changing environments [46,45]. Through a retrospective analysis on the use of the MUSA [33] middleware, the authors were able to identify the characteristics of a design approach for such a kind of systems.

The paper "*Stellar: A Programming Model for Developing Protocol-compliant Agents*" by Akin Günay and Amit Chopra presents the Stellar programming model to simplify development of protocol compliant agents. Stellar focuses on information-based interaction protocols, and provides a flexible event-driven programming model for the design and development of agents. To this end, Stellar defines a set of fundamental patterns and operations to facilitate the exchange of information among agents ensuring protocol compliance and eliminating common implementation errors. A major benefit of Stellar is its independence from imperative control flow structures, which gives substantial flexibility to developers when implementing agents compared to approaches that rely on such structure for compliance.

4.3 Formal analysis & techniques

The paper "*Slicing Agent Programs for More Efficient Verification*" by Michael Winikoff, Louise A. Dennis and Michael Fisher focuses on formal verification of agent programs using model checking. Formal verification of cognitive agents is highly desirable, since the complexity of their behavior makes assurance via traditional software testing infeasible [49,50]. However, current state-of-the-art techniques and tools for model checking cognitive agent programs are not able to deal with larger programs. This paper builds on a 2009 paper by Bordini *et al.* [7] which proposed to use *slicing*. The basic idea is to analyze a program prior to verifying it, and simplify the program by removing parts of it that cannot affect the result of verification of the property at hand. The paper defined an improved slicing method that was extended to handle features of a modern agent-oriented programming language, and that was more precise than the earlier slicing method.

Eleonora Giunchiglia presents an approach for "*Computing the Initial Requirements in Conditioned Behavior Trees*": her paper introduces an extension of Behavior Trees (BTs), a widely adopted model to represent single agent policies [12], called Conditioned Behavior Trees (CBTs). CBTs extend BTs because *(i)* they are able to represent policies in a multi-agent context and *(ii)* their actions are decorated with pre- and post- conditions. Further, the paper shows that given a CBT it is always possible to compute (in polynomial time) a propositional logic encoding whose models correspond to feasible plans. Hence, thanks to this encoding, the initial requirements can be easily obtained.

4.4 Rational agents techniques

Lukasz Bialek, Barbara Dunin-Keplicz and Andrzej Szałas introduce "*Belief Shadowing*". Adapting beliefs to new circumstances, like belief change, update, revision or merging, typically requires deep and/or complex adjustments of belief bases even when adaptations happen to be transient. The paper by Bialek *et al*., presents a lightweight and tractable approach to a new kind of beliefs' interference named belief shadowing, which takes place when part of one belief base is to be shadowed by another belief base representing new observations and/or beliefs of superior agents/teams. In this case no changes to belief bases are needed, and this substantially improves the performance of systems based on doxastic reasoning.
In the paper "*Resolving Incompatibilities among Procedural Goals under Uncertainty*", Mariela Morveli Espinoza, Juan Carlos Nieves, Ayslan Possebom and Cesar A. Tacla introduce a deliberative approach for dealing with conflicting goals in the settings of the practical reasoning. The suggested deliberative approach is based on formal argumentation theory such that plans are characterized by structured arguments. These structured arguments are measured by a novel

strength value defined by a three-dimensional vector determined from a probabilistic interval associated with each argument. The vector represents the precision of the interval, the location of it, and the combination of precision and location.

Timotheus Kampik, Juan Carlos Nieves and Helena Lindgren move a step "*Towards Empathic Autonomous Agents*". They explore the notion of an empathic autonomous agent that proactively searches for conflicts with other agents in its environment and employs a combined utility- and rule-based approach for resolving these conflicts. The authors propose an initial theoretical outline as well as a reasoning-loop architecture for their agent and highlight some challenges, in particular the handling of complex probabilistic environments and subjective observations.

The paper "*Intertemporal Equilibrium in Online Routing Games*" by László Zsolt Varga focuses on how to measure and ensure global behavior of large scale and open decentralized MAS. The paper shows how the intertemporal expectations of selfish planning agents influence the quality of the global behavior of the MAS in a realistic urban traffic scenario. The intertemporal expectations are derived from the information available to the agents, therefore the critical challenge is to design the environment in a way that drives the agents toward the optimum, or the equilibrium.

4.5 Modeling & simulations

Igor Conrado Alves de Lima, Luis Gustavo Nardin and Jaime Simão Sichman present "*Gavel: A Sanctioning Enforcement Framework*". Gavel enables agents to decide the most suitable sanctioning method, with the aim of improving agency governance. The framework is evaluated through a simulation of the Public Goods Game Model [15] with the CArtAgO [37] simulation framework.

In his paper "*Adding Organizational Reasoning to Agent-Based Simulations in GAMA*", John Bruntse Larsen discusses the importance of introducing organizational reasoning in a bottom-up agent platform such as GAMA, so that bottom-up BDI models and top-down organizational reasoning can be combined. The article formalizes the operational semantics of the organizational reasoning extension and illustrates its application with an example of healthcare. The purpose of introducing organizational reasoning in simulation is to model complex social systems where agents are organized and solve objectives for an organization while still being autonomous. Organizational reasoning does so by providing structure to an agent system, making roles, objectives, role dependencies, and obligations explicit and separate from the individual agents.

Tasio Méndez, J. Fernando Sánchez-Rada, Carlos A. Iglesias and Paul Cummings present "*A Model of Radicalization Growth using Agent-based Social Simulation*", where they propose a modeling approach of agent-based social simulator designed for modeling social networks, consisting of an Agent Model, for modeling the micro level and Network Model, for modeling meso and macro levels of analysis. This model has been implemented in an Agent Based Social Simulator and has been applied to Radicalism modeling, aiming at understanding radicalization roots as a first step for being able to define and applying suitable counter-terrorism measures.

4.6 Frameworks and application domains

Inga Rüb and Barbara Dunin-Kęplicz present a "*BDI Model of Connected and Autonomous Vehicles*": the search for balance between the huge complexity of representing real-world Connected and Autonomous Vehicles (CAVs) and comprehensibility of the solution suggests the adoption of a BDI approach. BDI systems offer useful abstractions for activities of a single self-driving car and a whole systems of such vehicles. The BDI framework also helps to combine two distinct features of a self-driving car: its reactiveness and proactiveness. Moreover, modularity of the resulting architectures for an individual CAV and urban traffic induced by these cars makes the design easily extensible and resilient.

In the paper "*Engineering World-Wide Multi-Agent Systems with Hypermedia*", Andrei Ciortea, Olivier Boissier and Alessandro Ricci propose an approach to engineer large-scale, evolvable MAS using hypermedia. In a *hypermedia MAS*, inline with the notion of agent environments [48], agents are situated in a distributed hypermedia environment that they can navigate and use in pursuit of their goals. Agents use the hypermedia to discover: (i) other entities in the MAS (e.g., other agents, tools, knowledge repositories, organizations, datasets etc.), and (ii) means to interact with those entities. This allows the MAS to evolve at runtime and to be seamlessly distributed across the Web. The authors report on a demonstrator in which BDI agents are able to use hypermedia to discover and interact with artifacts that are deployed at runtime and can evolve independently from the rest of the system.

"*Designing a Cognitive Agent Connector for Complex Environments: A Case Study with StarCraft*" by Vincent Koeman, Harm Griffioen, Danny Plenge and Koen Hindriks describes the design of a connector for the real-time strategy game StarCraft and its use as a case study for establishing a design method for developing connectors for environments. Connectors are a key element in MAS engineering because the evaluation of cognitive agent systems requires more benchmark environments that offer more features and involve controlling more units, and how to create a connector for interfacing these agents with such richer environments is still a challenging issue. Cognitive agents use knowledge technologies for representing state, their actions and percepts, and for deciding what to do next. StarCraft is particularly suitable as a testbed as it requires the design of complicated strategies for coordinating hundreds of units that need to solve a range of challenges including handling both short-term as well as long-term goals.

Maira Gatti de Bayser, Claudio Pinhanez, Heloisa Candello, Marisa Affonso Vasconcelos, Mauro Pichiliani, Melina Alberio Guerra, Paulo Cavalin and Renan Souza present "*Ravel: A MAS Orchestration platform for Human-Chatbots Conversations*". Ravel is a MAS aimed to integrate natural language understanding components with orchestration components of dialogues between human beings and agents. Ravel enables the specification of (social) conversations norms, using deontic logic, for use in contexts where multiple agents and human users are conversing in natural language. The usefulness of Ravel has been demonstrated in a chat-based finance adviser system designed as a chat group of five participants: four collaborative chatbots with two different roles (mediator and expert) and a human or chatbot user.

In the paper "*Human-Agent Interaction, the System Level Using JASON*", Antonio Chella, Francesco Lanza and Valeria Seidita discuss how to support the agent's decision process using the internal state. They propose an extension of the Jason reasoning cycle to deal with the implementation level of the decision process and to include elements coming from the internal state. This work is intended to meet challenges about knowledge representation and creation of plans at runtime.

Finally, Orso Negroni, Anthoni Othmani, Arthur Casals and Amal El Fallah-Seghrouchni present how "*Exposing Agents as Web Services in JADE*". The paper shows how intelligent agents using a BDI architecture can be exposed as web services and integrated with existing cloud services. This work consists of a Smart Agenda MAS built to function as an agent-based personal assistant. While exposing agents as web services is not a novel approach, the objective of this work is to understand (i) what is the current state of production-ready MAS, and (ii) how hard it is for a software developer to understand and implement an agents-based solution. For this reason, the Smart Agenda MAS was implemented by first-year graduate students (masters), which allowed the authors to observe how a MAS using very well-known Internet-

related paradigms can be modeled and implemented by developers that never had any contact with agent technologies. It was also possible to critically evaluate the learning curve involving the technologies used in the implementation (JADE [4] and BDI4JADE [29]).

5. WORKSHOP PANELS

5.1 Joint panel with the Goal Reasoning Workshop

Goals are a unifying structure across a variety of intelligent systems, and reasoning about goals takes many forms. In the most encompassing view, intelligent systems can use goal structures (or goal rewards) to manage long-term behavior, anticipate the future, select among priorities, commit to action, generate expectations, assess tradeoffs, resolve the impact of notable events, or learn from experience. As a result, the broad topic of goal reasoning is studied in diverse subfields of AI such as motivated systems, cognitive science, automated planning, and agent-oriented programming to name but a few. The Goal Reasoning Workshop (GRW)[2] brings together researchers to encourage cross-disciplinary discussion on goal reasoning. Given the strong connection between goal reasoning and MAS engineering, a joint panel with GRW was organized in the first day of the EMAS workshop, providing the opportunity to the attendees to address the topic of "*Requirements and Goals for Agent-based Systems: From Specification and Design to Runtime Representation and Reasoning*". The panelists Michael T. Cox, Koen Hindriks, Hector Munoz-Avila, M. Birna van Riemsdijk, and Michael Winikoff were moderated by Amal El Fallah-Seghrouchni. One conclusion was that each community can benefit by learning more about the work of the other community. For example, the GRW community has focused largely on single-agent settings, and could benefit from looking at the work on multi-agent concepts that have been developed in the EMAS (and more broadly AAMAS) communities (e.g. teams, roles, social commitments, norms). Another area where GRW could benefit from EMAS is, unsurprisingly, in Engineering aspects: methodologies, tools, and notations for developing agent systems. Conversely, the EMAS community could benefit by considering more deeply the question of goal adoption, and considering richer ways of dealing with the unexpected. It was also noted that the GRW community is predominantly North American in terms of both membership and research "style", and that in approaching EMAS work, which is more European in style, it is important to understand and appreciate the focus on formality and precision. Conversely, the EMAS community needs to understand and appreciate the focus on developing end-to-end solutions.

From a programming perspective, goals provide several benefits. They allow for potentially cleaner programs, for advanced goal reasoning, and goals allow for a very nice mapping to (classical) planning and thus provide for an interface between agent programming and planning. Goals in MAS programming have shown to have their use in avoiding interference between agents.

Key challenges that still need to be addressed include the addition of goals to a goal base, which is computationally more expensive than the addition of a belief to a belief base. Although both goals and beliefs may be modeled as facts in a Prolog-like setting, adding a new goal is more expensive than inserting a new belief, as in the second case there is no need for consistency checking (at least in such a database which only consists of positive literals), but some basic rationality constraints such as that the goal is not believed to be the case must be ensured in the first case. Also, goal management usually takes temporal aspects into account: as an example, "Alice may want to go to the library and to the movies", but this is typically meant in a temporal sense as she cannot be in two places at the same time.

[2] https://dtdannen.github.io/faim2018grw/

5.2 Community panel about EMAS papers & reviews

A community discussion about what kind of papers should be considered relevant and significant for EMAS and how the reviews should reflect this took place in the second day. The panelists were Andrei Ciortea, Eleonora Giunchiglia, Vincent Koeman, Tasio Mendéz, Juan Carlos Nieves, and László Zsolt Varga, moderated by Danny Weyns. The outcomes of the discussion are the following.

5.2.1 What is a good EMAS paper?

1. **Involved dimensions: technical vs theoretical; pure software engineering vs pure MAS.** EMAS papers can be characterized in two dimensions: one dimension is from technical towards theoretical paper, the other dimension is from pure software engineering towards pure multi-agent paper. Any paper in this two dimensional space may be a good EMAS paper if it points out *how certain properties of the system or the system creation process induces certain properties of the MAS*. As a typical EMAS paper will offer new programming, design, or verification techniques, such a paper needs to motivate and provide a rationale for the introduction of the technique and provide at least some validation of the usefulness or effectiveness of the technique.
2. **Engineering dimension.** By understanding engineering as the science concerned with putting scientific knowledge to practical uses, *a good EMAS paper introduces results that can help to design, implement and evaluate theories and methods of MAS*. By considering the practical use of MAS, a good EMAS paper shows clear use cases that motivate either the theoretical and practical use of MAS theories and models. A good EMAS paper should be based on solving real problems, as well as detail innovative experiences and new application domains. It should be also considered that the problem faced could be solved with MAS methods. A good EMAS paper should be able to analyze if MAS is a good solution for the problem we want to face.
3. **Reusability.** A good EMAS paper should contribute *reusable results*, in terms of theoretical insight, software and tooling, and validation.
4. **Connections with other communities.** A specific category of EMAS submissions that should be encouraged are those that *seek to create thorough conceptual and technological bridges with other communities*. Creating such bridges also implies finding research problems in other communities that can be addressed by reusing results from EMAS research. This would motivate and facilitate research transfer to other communities, therefore increasing the relevance of EMAS research outside of the AAMAS community.

5.2.2 What is a good EMAS review?

1. **Compliance with standard review guidelines.** An EMAS review is good if it follows general reviewer guidelines: it is courteous and constructive; it explains and backs up the judgment so that the authors are able to understand the reasoning behind the comments; it indicates whether the comments are opinions or are underpinned by facts; it shows that the reviewer understood the research; and it states whether it has a sufficient impact and adds to the knowledge base. A good EMAS review always avoids harassment issues.
2. **Compliance with the EMAS Call for Papers.** A good EMAS review recognizes the positive and negative aspects of a paper, which *should be written according to the recommendations of the EMAS Call for Papers*.

3. **Maieutic approach.** A good EMAS review should be constructive and should provide hints for improvement, *but not give the solution*. It should be the most explicit and detailed about each of the aspects and provide suggestions on how they could solve each of the problems presented. A good review supports the learning processes of the student co-authors. This support can be done by the suggestion of references of the state of the art, posting questions that help the student co-authors to realize about the particular research tracks of the EMAS community.

6. WORKSHOP DISCUSSION GROUPS

The second day of the workshop was mainly devoted to discussion. The co-chairs presented the discussion topics and asked the audience to add topics if they wanted. The people in the room selected a topic of their choice in such a way to create groups with at least 4 persons, and no more than 7. After two sessions when groups worked on the selected topic, each group presented the result of the discussion, reported below.

6.1 Cognitive Agent Architectures

Participants: Rem Collier, Louise Dennis, Lars-Åke Fredlund, Vincent Koeman, Sam Leask, Brian Logan, Juan Carlos Nieves.

6.1.1 Why is the topic important?

Although cognitive agent architectures have been studied for decades, with particularly rapid development in the area of agent programming languages around 15 years ago [36,6] it is far from clear that a satisfactory communal understanding of these languages has been reached: particularly with respect to their core features and inherent limitations. It was the view of the group that potentially too quick a convergence on a core functionality had been achieved. This meant that application of the languages could prove more challenging than necessary and that important areas of theory had been omitted [18].

6.1.2 What are the challenges?

1. **Programming plans or rules that are applicable in all situations is very hard.** Plan contexts typically used to control applicability are a key source of bugs in agent programming [50] and the need to construct contexts to cover all eventualities results in programs containing a proliferation of plans and a concomitant reduction in the transparency of code - contrary to an off-stated assumption that agent programming languages encourage a declarative and understandable programming style. The issue typically manifests as control over the selection of plans, but is also relevant to the selection of goals and the selection and scheduling of intentions (two areas that have received less attention from the community though they have not been entirely ignored, see for instance [34,43,44,52]). This is also a key problem that has hindered the ability of agent programming languages to come up with a coherent and widely agreed upon framework to support modular programming and so enable reuse.
2. **How can interaction be well-engineered in open multi-agent systems?** When interactions between agents are considered all aspects of program development (verification, data integrity, resource management and concurrency) become much harder [22]. This appears to have contributed to a lack of strong links between the agent programming community and the game theory community that makes up a large and apparently separate part of the autonomous agent landscape. It was noted that programming frameworks had been developed for engineering cognitive agent interactions (most notably the MOISE+ framework [19]), particularly frameworks exploiting ideas of organizations and roles, but accounts of their interactions with core agent programming concepts were more ad hoc.

6.1.3 What are the particular action points for EMAS community? AAMAS community? SE community?

The EMAS community needs to avoid falling into the assumption that the theoretical questions around the nature and core concepts of cognitive agent architectures are solved and fixating too heavily around questions relating to large scale deployment and the development of industry-strength tools.

A large number of tools and theories have been generated by other communities that may well have significant contributions to make to the further development of cognitive agent architectures. For instance the Coordination, Organizations, Institutions and Norms (COIN) community, Answer Set Programming, Argumentation, Game Theory, AI Planning and Knowledge Representations communities all potentially have a contribution to make to the challenges highlighted above.

Given this observation it is important to investigate the integration between cognitive agent frameworks and other decision-making/reasoning models which is far from trivial. Hence, there is a need to investigate “dynamic” cognitive agent architectures that allow an agent to have multi-modal reasoning methods. An important additional motivation here (not highlighted above) is the huge variety of data that a real agent/MAS has to handle.

The group was keen to highlight that such integrative work should be undertaken with a view to creating a coherent architecture and not on developing ad hoc solutions to specific problems.

There is also scope for work on mapping the space of agent programming languages, specifically in understanding what they are and what they are not, since these languages are notable for an assumption that they are embedded in larger software systems with some aspects of computation delegated outside the agent.

6.2 AgentSpeak++

Participants: John Bruntse Larsen, Angelo Ferrando, Julian Padget, Alessandro Ricci, Michael Winikoff.

6.2.1 Why is the topic important?

Programming languages are crucial tools for expressing and realizing software systems. The right programming language can make a huge difference to the ease of writing (and modifying) software. Accordingly, it is hardly surprising that there has been substantial work on specialist programming languages for developing multi-agent systems.

However, agent-oriented programming has not caught on, and one argument for why this is the case [23] is that Agent-Oriented Programming Languages (AOPLs) fail to provide the right balance of features: they leave too much to the programmer to specify, so that the overall benefit does not, in many cases, outweigh the learning cost.

The diffusion of MAS applications in the real world has not reached its full potential yet. Reasons for this can be found for example in [47,30]:

1. limited awareness about the potential of agent technologies;
2. limited publicity of successful industrial projects carried out with agent technologies;
3. misunderstandings about the effectiveness of agent-based solutions;
4. risks of adopting a technology that has not been already proven in large scale industrial applications;
5. lack of mature enough design and development tools for industrial deployment;
6. lack of integration with common engineering practice.

Specifically, there are many features that are not adequately addressed in current state-of-the-art AOPLs, such as:

- lookahead planning;
- learning;
- more intelligent decision making taking into account priorities, costs, and interactions between goals and between intentions;
- reasoning about when to adopt and when to drop goals; and
- dealing with open systems;
- embedding agent-based software engineering in mainstream software engineering practice.

6.2.2 What are the challenges?

The challenge is to develop AOPLs that provide richer reasoning abilities, and that do so in a way that is:

1. easy for the programmer to learn;
2. easy for them to use (including being easy to understand what is going on); and
3. that, overall, results in a reduction in the programming effort required, by allowing the programmer to leave certain aspects to the language (analogy: in a modern garbage collected programming language the programmer does not worry about manually managing memory).

In order to achieve these aims, which are in tension with each other (e.g. a more expressive and powerful notation will in general be harder to understand and to learn), we need to consider whether the result should be best realized as a programming notation, or as a library, pattern, or service. Additionally, of course, tools need to be robust, work, be documented, and support all development activities.
One approach that can help is to have a modular decision making process. This can allow the programmer to consider only those aspects of the decision-making process that are relevant to a particular application.
Another interesting aspect in the development of MAS is their verifiability. There exist many different approaches to check the consistency of agents behavior with respect to a given property. Both from a static and a dynamic viewpoint. One of the important aspects that could be interesting to explore and extend in future works concerning the development of new AOPLs is to find new ways to integrate the verification inside the AOPLs process, trying to develop self-contained verification mechanisms that support a non invasive way to develop more robust MAS.

6.2.3 What are the particular action points for EMAS community? AAMAS community? SE community?

The group proposed the following actions as concrete steps forward.

- In order to evaluate progress (and to aid in developing new notations) we need a collection of cases and applications. This would allow new or modified notations to be assessed to see whether they can be used to express a range of applications, and to assess to what extent they improve on existing notations on a broad range of problems. It is worth noting that "better" here can relate to a range of criteria: easier to write, easier to modify, easier to explain, easier to verify, the notation being easier to use, being more efficient (run-time), and, perhaps more importantly, the amount of code that a given notation allows the developer to avoid writing.
- In order to provide a simple conceptual foundation for richer decision making, the group suggests that perhaps the time has come to move away from AgentSpeak-style reactive plans, and instead use Hierarchical Task Network-style recipes [41]. This would support (limited) lookahead planning and evaluation to select a course of action, and this selection could also perhaps be modularly (and gradually) extended to incorporate (e.g.) priorities, resources, values in selecting between alternative options.

There is a need for the community to work towards a consensus on a simple set of concepts for social aspects: there are various models (e.g. AORTA [39], MOISE [19]) that can be exploited to this aim. As a concluding remark, the group posed the question: is there now enough experience and agreement to develop a single model for social concepts that is both general, and powerful?

6.3 Machine Learning & MAS

Participants: Eleonora Giunchiglia, Timotheus Kampik, Tasio Mendez, Zahia Guessoum, Danny Weyns.

6.3.1 Why is the topic important?

New information systems and recent applications are often distributed, large scale, open, heterogeneous and deployed to dynamic environments. To model these complex systems, much research effort was spent during the last years on MAS. To reach their goals in such dynamic and changing environments, many researchers have highlighted the need to use machine learning (ML) to build adaptive agents and MAS. The focus was on using or creating new learning paradigms for MAS to design and control complex systems. Several topics were proposed by the Adaptive Learning Agents workshop[3] such as :

- Integrating learning to build opponent models in negotiation, trust models, coordination, etc.
- Reinforcement learning (single and multi-agent).
- Distributed learning.
- Adaptation and learning in dynamic and complex environments.
- Design of reward structure and fitness measures for coordination.
- Scaling learning techniques to large systems of learning and adaptive agents.

Another interesting topic is the use of learning to improve the design of MAS. For example, several approaches have been proposed to select the most efficient organization of agents or the learning to improve the design of the agents' environment [48].
While the increase in the availability of computing resources, as well as the increasingly creative application of learning techniques has led to some promising advancements in the application of ML, in pa rticular in speech and image recognition, deep learning methods have some severe limitations, for example as they typically lack transparency and do not integrate well with prior knowledge [24].
Given the opportunities ML offers, as well as its current limitations, MAS can both benefit from applying ML, and contribute to enhancing ML methods.

Learning for MAS is a topic that is generally covered by existing literature (see for example [2]). However, despite its limitations, recent advances in (deep) learning techniques can be expected to address some challenges MAS faces related to handling large state spaces in dynamic and open environments, for example by applying deep reinforcement learning [28]. In addition, the MAS community can learn from the ML community's success in creating tools and methods that are relevant to and widely adopted by software engineering practitioners. At least some research that employs MAS to enhance ML is already emerging. For example, Irving *et al.* employ multi-agent argumentation to debate the plausibility of ML results [20]. Still, it can be expected that there are research opportunities for MAS experts that have not been explored, yet.

6.3.2 What are the challenges?

The following challenges of combining MAS with machine learning have been identified:

[3] ALA: http://ala2018.it.nuigalway.ie/

1. **While ML can be a powerful problem solver, it typically does not provide the guarantees MAS requires.** MAS as an engineering discipline often requires guarantees that can be formally verified. Applying models generated by (deep) learning methods typically does not allow yet for such guarantees. Hence, a challenge is to determine in what cases such guarantees can be relaxed (replaced by "soft" guarantees) and if and how ML can be applied without weakening guarantees.
2. **While the MAS and ML communities have intersections, not enough exchange takes place.** Currently, machine learning, and in particular deep learning, is a topic at the center of attention in both academia and industry. As a consequence, many university graduates, as well as experienced industry practitioners, join the ML community to become researchers. The spotlight on ML makes it harder for other AI communities to attract talent and also increases the likelihood that members of the ML community are ignorant of relevant research in MAS and other subfields.
3. **MAS and ML use different technological ecosystems.** Many frameworks that are frequently used by the MAS community - for example Jason and JaCaMo - have not widely been adopted in practice and are dependent on technologies that are losing traction in the industry. In contrast, technologies that are popular among members of the ML community, like TensorFlow [1], Keras[4], and OpenAI Gym[5] are increasingly popular among software engineering practitioners. It is up to the MAS community to bridge this gap to have an impact outside of academia.

6.3.3 What are the particular action points for EMAS community? AAMAS community? SE community?

On the one hand, the EMAS community can embrace the recent advances in ML and:

- Identify MAS problems that can be potentially addressed with ML techniques injected into standard MAS frameworks and device corresponding frameworks and good practices;
- Develop performance benchmarks for MAS-specific ML problems;
- Invite members of the ML community to contribute to the EMAS community.

On the other hand, members of the EMAS community can contribute directly to ML research by:

- Identifying ML problems that can potentially be addressed with multi-agent systems techniques and work towards MAS-based solutions and frameworks;
- Building technology bridges from traditional MAS concepts like BDI agents to ML frameworks that are at the bleeding edge of applicable technology;
- Presenting ML-related MAS research at ML venues.

6.4 Cognitive agents and MAS programming

Participants: Arthur Casals, Andrei Ciortea, Amal El Fallah-Seghrouchni, Viviana Mascardi, Valeria Seidita, László Zsolt Varga.

6.4.1 Why is the topic important?

Finding boundaries between AOSE approaches and other SE methods: AOSE is a very powerful approach to understand, describe and design complex systems, although agents are not necessarily implemented into the final systems in their pure theoretical form, and using purely agent-oriented languages like Jason, 3APL, etc. Rather, agents may be built into applications by exploiting tools and languages which are specific to the given application domain, and which prove more suitable than standard agent programming languages in that specific context. A better understanding of the relationships between AOSE stages and more widespread and consolidated SE approaches/tools/languages would allow AOSE and agent technology to be more easily accepted for the engineering stages where they give concrete advantages, and complemented with other domain-specific solutions when they better fit the developers' needs.

Finding boundaries between applications that do need agents, and applications that do not need them: Also, not all the applications require to be engineered following an AOSE approach: think for example of monolithic and centralized applications, that do not need to be aware of the surrounding environment. Despite their relevance, flexibility and capabilities, agents are not necessarily meant to be used everywhere. Although dating back to almost twenty years ago, the survey by Stone and Veloso [35] provides answers to the questions "What advantages agent technology offers over the alternatives?" and "In what circumstances is it useful?" that are still valid in the current scenario, and still worth reading. They write that

> "*Like any useful approach, there are some situations for which it is particularly appropriate, and others for which it is not. [...] In particular, if there are different people or organizations with* ***different (possibly conflicting) goals and proprietary information****, then a multiagent system is needed to handle their interactions. [...] Having multiple agents could speed up a system's operation by* ***providing a method for parallel computation****. [...] While parallelism is achieved by assigning different tasks or abilities to different agents,* ***robustness*** *is a benefit of multiagent systems that have redundant agents. If control and responsibilities are sufficiently shared among different agents, the system* ***can tolerate failures*** *by one or more of the agents. [...] Another benefit of multiagent systems is their* ***scalability****.*
>
> *From a programmer's perspective, the* ***modularity*** *of multiagent systems can lead to simpler programming. [...] Finally, multiagent systems can be useful for their* ***illucidation of intelligence.***"

The features mentioned by Stone and Veloso make agents and MAS programming appealing for facing the engineering challenges raised by nowadays complex applications, including the "Internet of Smart Things": IoT systems which must be resilient, efficient, and "smart" [26,31]. When agents are cognitive ones, the ability to explicitly model themselves, other agents, and the surrounding environment, and to reason about these models, can be exploited to pursue machine ethics[6] also following some ethics-aware software engineering approach [3]. Other issues should be explored for understanding (inside the EMAS/AAMAS communities) and explaining (outside them) in which domains cognitive agents and MAS programming can be applied and used to their full extent, finding where the application boundaries are and allowing the technology's potential to be fully explored. Another related pointer that discusses these matters is [47].

Bridging: besides finding the boundaries between applications that can benefit from being engineered following an agent-oriented approach, and those that cannot, it is also important to explore the intersection between agent technologies and other different paradigms, such as Web of Things (Section 3) and Machine Learning (Section 6.3). It is possible that some of the solutions proposed for problems within these domains can have better results when used in conjunction with MASs. A practical example of bridging agents with other paradigms was shown in [10], and in his keynote presentation at AAMAS 2018, Tenenbaum observes that "*human intelligence is more than just pattern recognition. And no machine system yet built has anything like the flexible, general-purpose commonsense grasp of the world that we can see in even a one-year-old human infant*" [42]. There is no one technology alone that can

[4] Keras: The Python Deep Learning library, https://keras.io/

[5] A toolkit for developing and comparing reinforcement learning algorithms, https://gym.openai.com/

[6] Asilomar AI Principles, https://futureoflife.org/ai-principles/

achieve the goal of creating *really* intelligent agents: cross-fertilization is the key for success.

6.4.2 What are the challenges?

The challenges related to expanding the sphere of influence of agents technologies are closely connected with the reasons for which the topic is important. They can be summarized in:
1. Finding problems or application domains where we can do something new or something better: as a consequence, companies that see that agents can actually solve their problems, would start using these technologies.
2. Reaching and working in conjunction with other communities in order to make them aware of what we did in the last 30 years, and to transferring our knowledge to them (social challenge).
3. Building the conceptual bridges and the technological bridges that we need to make cross-fertilization and interoperation among different communities possible (research and technological challenge).

These are not separated challenges, but steps that belong to the same effort: in particular, 1 comes before 2, which in turn comes before 3.

6.4.3 What are the particular action points for EMAS community? AAMAS community? SE community?

W.r.t. this issue, the discussion group devised very concrete actions, all aimed at allowing neighboring research communities including SE, IoT, Semantic Web, Security, Cyberphysical systems, etc, to know about the EMAS/AAMAS community, and us know about them:
1. Foresee an "Interesting topics which could fit the EMAS goals and needs, but are not explicitly tagged as agent-oriented" track at EMAS 2019 expressly designed to push cross-fertilization among research communities;
2. Invite keynotes from other communities (which is maybe more interesting for the EMAS community, than for cross-fertilization purposes);
3. Disseminate our findings by submitting papers to conferences and journal in different but neighboring areas; some examples include the IoT conference and related events[7] and TOSEM[8]/TSE[9] journals.

7. CONCLUSIONS

The goal of engineering processes is to design and implement systems that meet the design criteria, and to show that the final product actually satisfies the specifications. Current software engineering practices focus mainly on the creation process (technical-engineering aspect) in the hope that good practices and tools lead to good products, because the theoretical approach is considered too complex. However, a good design needs strong theoretical background to select the best architectural and algorithmic design options. This is even more important for engineering MAS, because ensuring the global behavior of large-scale MAS includes not only implementing agents that reach their goals, but also designing how the environment, the agents, the organization of the agents can interact with each other (theoretical-MAS aspect) in order to optimize their global behavior.
In order to have agents being widely (and appropriately) used in real-world applications, it is important to understand which of the Industry problems can actually be solved (or benefit) from agents (sections 6.4.1, 6.4.2). For the same reason, it is also important to coordinate an effort of cross-pollination between the agents community and other communities that (i) can benefit from agents in their existing solutions and (ii) are closer to the Industry environment and its real problems. Finally, all the above efforts might turn out to be useless if the EMAS goal of involving more master and PhD students would not succeed. Understanding real-world problems, bridging with other research communities and with the Industry, and creating good practices and tools should be accompanied by a parallel effort of involving students, in particular PhD ones. PhD students have in fact time to explore and adapt general tools to their specific purposes, while MSc students need off-the-shelf tools to solve broader problems in a shorter period of time. Both of them can give precious feedback in using or improving existing technologies, and both are usually composed by curious, "fresh", open-minded persons. Because of their lack of experience in the MAS setting (or better, thank to it), students can point out issues, connections, threats and opportunities that a more "senior" community might not notice any longer. In the EMAS 2018 edition, the involvement of students as panelists showed the impressive potential that they have, even when they have a very limited technical background on agents and MASs. If we want to change the way agent technology is perceived outside the EMAS/AAMAS community, we have to invest resources and efforts in training the youngs. After all, "*education is the most powerful weapon which you can use to change the world*" (Nelson Mandela).

[7] https://www.iotevents.org/
[8] https://tosem.acm.org/
[9] https://www.computer.org/web/tse/

8. ACKNOWLEDGMENTS

Arthur Casals is supported by CNPq, grant no. 142126/2017-9.
Eleonora Giunchiglia is supported by the EPSRC, under grant EP/M508111/1, and the Oxford-DeepMind Graduate Scholarship.
Timotheus Kampik is supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.
Geylani Kardas, Moharram Challenger and Baris Tekin Tezel are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant 115E591.
Simon Mayer is supported by the Austrian FFG under grant #854184 (COMET).
The work of László Z. Varga was carried out in the project EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies - The Project is supported by the Hungarian Government and co-financed by the European Social Fund.

9. REFERENCES

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. and Zheng, X. 2016. TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265-283.

[2] Alonso, E., D'inverno, M., Kudenko, D., Luck, M. and Noble, J. 2001. Learning in multi-agent systems. The Knowledge Engineering Review, 16(3), pp. 277-284.

[3] Aydemir, F. B. and Dalpiaz, F. 2018. A roadmap for ethics-aware software engineering, In Proceedings of the International Workshop on Software Fairness, FairWare@ICSE, pp. 15-21, ACM.

[4] Bellifemine, F. L., Caire, G., and Greenwood, D. 2007. *Developing multi-agent systems with JADE*. John Wiley & Sons

[5] Boissier, O., Bordini, R.H., Hübner, J., Ricci, A. and Santi, A. 2013. Multi-agent oriented programming with JaCaMo. Science of Computer Programming, 78(6), pp.747–761.

[6] Bordini, R. H., Hübner, J. F. and Wooldridge, M. 2007. Programming multi-agent systems in AgentSpeak using Jason. John Wiley & Sons.

[7] Bordini, R. H., Fisher, M., Wooldridge, M. and Visser, W. 2009. Property-based slicing for agent verification. J. Log. Comput. 19(6), pp. 1385-1425.

[8] Challenger, M., Demirkol, S., Getir, S., Mernik, M., Kardas, G., and Kosar, T. 2014. On the use of a domain-specific modeling

language in the development of multiagent systems. Engineering Applications of Artificial Intelligence, 28, pp. 111-141.

[9] Challenger, M., Kardas, G. and Tekinerdogan, B. 2016. A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems. Software Quality Journal, 24(3), pp. 755-795.

[10] Ciortea, A., Boissier, O., Ricci, A. 2017. Beyond physical mashups: Autonomous systems for the Web of Things. In Proceedings of the Eighth International Workshop on the Web of Things, WoT 2017, pp. 16-20, ACM.

[11] Ciortea, A., Mayer, S. and Michahelles, F. 2018. Repurposing manufacturing lines on the fly with multi-agent systems for the Web of Things. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (AAMAS '18). International Foundation for Autonomous Agents and Multiagent Systems, 813-822.

[12] Colledanchise, M. and Ögren, P. 2017. How behavior trees modularize hybrid control systems and generalize sequential behavior compositions, the subsumption architecture, and decision trees. IEEE Trans. Robotics 33(2), pp. 372-389.

[13] Collier, R.W., Russell, S.E. and Lillis, D. 2015. Reflecting on agent programming with AgentSpeak(L). In Proceedings of PRIMA 2015: Principles and Practice of Multi-Agent Systems - 18th International Conference. Lecture Notes in Computer Science, vol. 9387, pp. 351–366. Springer.

[14] Fisher, M., and Ghidini, C. 2010. Executable specifications of resource-bounded agents, Journal of Autonomous Agents and Multi-Agent Systems, 21(3), pp. 368-396.

[15] Giardini, F., Paolucci, M., Villatoro, D. and Conte, R. 2014. Punishment and gossip: Sustaining cooperation in a public goods game. In Proceedings of Advances in Social Simulation. Advances in Intelligent Systems and Computing, vol. 229, pp. 107–118. Springer.

[16] Hindriks, K. V., de Boer, F. S., van der Hoek, W. and Meyer, J.-J. 1999. Agent programming in 3APL, Journal of Autonomous Agents and Multi-Agent Systems, 2(4), 357-401.

[17] Hindriks, K. V., de Boer, F. S., van der Hoek, W. and Meyer, J.-J. 2001. Agent programming with declarative goals, In Proceedings of Intelligent Agents VII, 6th Workshop on Agent Theories, Architectures, and Languages, LNAI, vol. 1986, pp. 228-243. Springer.

[18] Hindriks, K. V. 2014. The shaping of the agent-oriented mindset. In Engineering Multi-Agent Systems. pp. 1-14. Springer.

[19] Hübner, J. F., Boissier, O., Kitio, R., and Ricci, A. 2010. Instrumenting multi-agent organisations with organisational artifacts and agents. Journal of Autonomous Agents and Multi-Agent Systems, 20(3), pp. 369-400.

[20] Irving, G., Christiano, P. and Amodei, D. 2018. AI safety via debate. Preprint available at arXiv:1805.00899.

[21] Kardas, G. and Gomez-Sanz, J. J. 2017. Special issue on model-driven engineering of multi-agent systems in theory and practice. Computer Languages, Systems & Structures, 50, pp. 140-141.

[22] Koeman, V. J., Hindriks, K. V. and Jonker, C. M. 2017. Designing a source-level debugger for cognitive agent programs. Journal of Autonomous Agents and Multi-Agent Systems, 31(5), pp. 941-970.

[23] Logan, B. 2018. An agent programming manifesto. International Journal of Agent-Oriented Software Engineering, 6(2), pp. 187-210.

[24] Marcus, G. 2018. Deep learning: A critical appraisal. Preprint available at arXiv:1801.00631.

[25] Mayer, S., Verborgh, R., Kovatsch, M. and Mattern, F. 2016. Smart configuration of smart environments. IEEE Trans. Automation Science and Engineering 13(3): 1247-1255.

[26] van Moergestel, L., van den Berg, M., Knol, M., van der Paauw, R., van Voorst, K., Puik, E., Telgen ,D. and Meyer, J. 2016. Internet of Smart Things - A study on embedding agents and information in a device. In Proceedings of the 8th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART, pp. 102-109.

[27] Moody, D. 2009. The “Physics” of Notations: Toward a scientific basis for constructing visual notations in software engineering, IEEE Transactions on Software Engineering, 35(6), pp. 756-779.

[28] Mousavi, S. S., Schukat, M. and Howley, E. 2016. Deep reinforcement learning: an overview. Proceedings of SAI Intelligent Systems Conference, pp. 426-440, Springer.

[29] Nunes, I., De Lucena, C. J. P. and Luck M. 2011. BDI4JADE: a BDI layer on top of JADE. Presented at the 9th International Workshop on Programming Multi-Agent Systems, ProMAS 2011.

[30] Pechoucek, M. and Marík, V. 2008. Industrial deployment of multi-agent technologies: review and selected case studies. Journal of Autonomous Agents and Multi-Agent Systems, 17(3), pp. 397-431.

[31] Pico-Valencia, P. and Holgado-Terriza, J. H. 2018. Agentification of the Internet of Things: A systematic literature review, International Journal of Distributed Sensor Networks, 14(10).

[32] Pokahr, A., Braubach, L. and Lamersdorf, W. 2005. Jadex: A BDI Reasoning Engine. In Multi-Agent Programming. Multiagent Systems, Artificial Societies, and Simulated Organizations (International Book Series), vol 15. Springer.

[33] Sabatucci, L., Lopes, S. and Cossentino, M. 2017. MUSA 2.0: A distributed and scalable middleware for user-driven service adaptation. In Proceedings of the International Conference on Intelligent Interactive Multimedia Systems and Services, pp. 492-501, Springer.

[34] Shaw, P. H. and Bordini, R. H. 2010. An alternative approach for reasoning about the goal-plan tree problem. In Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010), vol. 215, pp. 1035-1036, IOS Press.

[35] Stone, P., Veloso, M. M. 2000. Multiagent systems: A survey from a machine learning perspective. Auton. Robots 8(3), pp. 345-383.

[36] Rao, A.S. 1996. AgentSpeak(L): BDI agents speak out in a logical computable language. In Proceedings of Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World. Lecture Notes in Computer Science, vol. 1038, pp. 42–55, Springer.

[37] Ricci, A., Piunti, M., Viroli, M., Omicini, A. 2009. Environment programming in CArtAgO. In Multi-Agent Programming: Languages, Tools and Applications, pp. 259–288. Springer.

[38] Russell, S. J. and Norvig, P. 2009. *Artificial Intelligence: A modern approach* (3rd ed.). Pearson Education.

[39] Schmidt Jensen, A., Dignum, V. and Villadsen, J. 2014. The AORTA architecture: integrating organizational reasoning in Jason. In Proceedings of EMAS@AAMAS 2014, pp. 127-145.

[40] Sierhuis, M. 2001, Modeling and simulating work pratice. BRAHMS: a multiagent modeling and simluation language for work system analysis and design, Ph.D. Thesis, Social Science and Informatics (SW), University of Amsterdam.

[41] Tate, A. 1976. Project planning using a hierarchic non-linear planner, D.A.I. Research Report No. 25, August 1976, Department of Artificial Intelligence, University of Edinburgh

[42] Tenenbaum, J. 2018. Building machines that learn and think like people. In Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), IFAAMAS.

[43] Thangarajah, J. and Padgham, L. 2011. Computationally effective reasoning about goal interactions, Journal of Automated Reasoning, 47(1), pp. 17-56.

[44] Waters, M., Padgham, L., and Sardina, S. 2015. Improving domain-independent intention selection in BDI systems, Journal of Autonomous Agents and Multi-Agent Systems, 29(4), pp. 683-717.

[45] Weyns, D. 2018, Engineering Self-Adaptive Systems. In Handbook of Software Engineering. Eds. C. Sungdeok, R. Taylor, K. Kang. Springer.

[46] Weyns, D. and M.P. Georgeff. 2010. Self-Adaptation Using Multiagent Systems. IEEE Software 27(1), pp. 86-91.

[47] Weyns, D. Helleboogh, A. and Holvoet, T. 2009. How to get multi-agent systems accepted in industry? International Journal on Agent-Oriented Software Engineering, IJAOSE, 3(4).

[48] Weyns, D. and Michel. F. 2014. Agent environments for multi-agent systems --- A research Roadmap. In Revised Selected and Invited Papers of the 4th International Workshop on Agent Environments for Multi-Agent Systems IV, Vol. 9068 Springer.

[49] Winikoff, M. and Cranefield, S. 2014. On the testability of BDI agent systems. J. Artif. Intell. Res., 51, pp. 71-131.

[50] Winikoff, M. 2014. Novice programmers' faults & failures in GOAL programs, In Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, pp. 301-308.

[51] Winikoff, M. 2017. BDI agent testability revisited. Autonomous Agents and Multi-Agent Systems, 31(5), pp. 1094-1132.

[52] Yao, Y. and Logan, B. 2016. Action-level intention selection for BDI agents. In Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), pp. 1227-1235.